Impact of Considering the Effects of Latent Heat on Caldera-Hosted Hydrothermal Systems

Melissa Barnard, BSc, BEd

Master of Science in Earth Sciences

Submitted in partial fulfillment of the requirements for the degree of

Master of Science

Faculty of Mathematics and Science, Brock University
St. Catharines, Ontario

ABSTRACT

To investigate the thermal effects of latent heat in hydrothermal settings, an extension was made to the existing finite-element numerical modelling software, Aquarius. The latent heat algorithm was validated using a series of column models, which analysed the effects of permeability (flow rate), thermal gradient, and position along the two-phase curve (pressure). Increasing the flow rate and pressure increases displacement of the liquid-steam boundary from an initial position determined without accounting for latent heat while increasing the thermal gradient decreases that displacement. Application to a regional scale model of a caldera-hosted hydrothermal system based on a representative suite of calderas (e.g., Yellowstone, Creede, Valles Grande) led to oscillations in the model solution. Oscillations can be reduced or eliminated by mesh refinement, which requires greater computation effort. Results indicate that latent heat should be accounted for to accurately model phase change conditions in hydrothermal settings.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF QUANTITIES AND SUBSCRIPTS/SUPERSCRIPTS

QUANTITIES

α	Compressibility of rock matrix (1/Pa)
β	Compressibility of fluid (1/Pa)
C	Specific heat capacity (J/kg °C)
D	Dispersivity (m)
δ	Dirac Delta function
g	Gravitational constant (m/s^2)
h	Equivalent freshwater head (m)
ΔH	Enthalpy (J/kg)
K	Hydraulic conductivity (m/s)
k	Permeability tensor (m^2)
λ_{ij}	Conduction-dispersion tensor
λ_f	Thermal conductivity (W/m °C)
P	Fluid pressure (MPa)
Q	External rates of heat supply (i.e. heat sources or sinks) (W/m)
q	Volumetric fluid flux or specific discharge (m/s)
Pe	Peclet number
ρ	Density (kg/m^3)
T	Temperature (°C)
t	Time (s)
φ	Porosity or pore fraction of medium (dimensionless)
μ	Kinematic viscosity of fluid (Pa-s)
z	Elevation at which freshwater head is calculated (m)

SUBSCRIPTS/SUPERSCRIPTS

f	Fluid
i	Coordinate direction
j	Coordinate direction
s	Solid
x	Coordinate direction
z	Coordinate direction

1. Introduction

Hydrothermal convection systems result when igneous rocks intrude relatively cooler, porous and permeable rocks at shallow levels within the Earth's crust. Such intrusion occurs in association with volcanic activity in zones of subduction, regions of plate stretching or spreading, and "hot spots" or mantle plumes. Hydrothermal convection occurs because the fluids, primarily water, heated by the intrusion are less dense than fluids in surrounding rocks, causing the heated fluids to rise through the crust away from the intrusion until they cool and descend again [Barnes, 1997; Energy and Geoscience Institute (EGI), 2001; Turcotte & Schubert, 1982].

Steam formation is an important process in many hydrothermal systems and was the focus of this study. This study involved the development and application of an algorithm to account for the associated thermal effects of latent heat. This algorithm was added to the Aquarius modelling software designed by Dr. Stephen J. Cook and successfully applied to the reconstruction of the hydrothermal flow system responsible for contact metamorphism and ore deposition surrounding the Alta Stock in Utah's Little Cottonwood Mining District [Bowman, Willett, & Cook, 1994; Cook & Bowman, 1994, 2000; Cook, Bowman, & Forster, 1997]. The ability to model phase change conditions accurately and to understand the factors that control steam formation and distribution is essential for determining effective methods of locating and preserving or exploiting resources associated with modern or active and ancient hydrothermal systems.

In active hydrothermal systems steam plays an important role in creating geysers and associated phenomena such as those found in the United States' Yellowstone system, Iceland's vent fields, and New Zealand's Kermadec Island Arc. Active hydrothermal

systems can and have been used as a source of geothermal power, which has fewer negative impacts on the planet because geothermal power generation facilities require less land, release negligible amounts of gases into the atmosphere, and allow residual waters to be re-injected into the geothermal reservoir [EGI, 2001].

Since 1904, geothermal experiments and power production have occurred in Larderello, Italy [International Geothermal Association (IGA), 2009]. The single large hydrothermal system below this site is recharged by meteoric waters, which are superheated by deep magmatic sources [Della Vedova et al., 2007]. Since the 1970's the largest producer of geothermal electrical power has been The Geysers Geothermal Field in California where the first system using wastewater to generate electricity was later implemented in 1997 to enhance productivity and longevity of The Geysers after reservoir pressure dropped [Sass & Priest, 2002].

Steam formation is also an established mechanism for ore formation in a variety of epithermal ore deposits [Barnes, 1997]. Boiling or effervescence (the separation of gas from an aqueous solution) can result from decreases in pressure as hydrothermal fluids circulate through the Earth's crust. In turn, solution concentration is increased and volatiles are removed from the solution. Volatile removal leaves the solution more alkaline and less capable of transporting metals [Barnes, 1997; Robb, 2005]. When those metals precipitate in large enough quantities in a localized region, an ore deposit is formed.

Bingham Canyon in Utah, USA is the location of a porphyry copper deposit made by circulating hydrothermal fluids, which altered the surrounding rock [Edwards & Atkinson, 1986]. Copper, gold, silver, and molybdenum are all at the large open-pit mine

at this site [InfoMine Inc, 2008; Rio Tinto, 2008]. At Olympic Dam in Australia ore deposits resulted when hot fluids mixed with cooler meteoric fluids [Robb, 2005]. Mining at this location is the world's largest uranium production and fourth largest copper production [InfoMine Inc., 2008]. Such valuable ores are of great economic importance because of their useful chemical and physical properties.

In active hydrothermal systems, direct observation of the features at and below the Earth's surface is hindered by the high pressures in the system and the corrosive nature of hydrothermal fluids [Bryan, 2001]. Geothermal reservoirs are hidden beneath these regions. Ore deposits are discovered at ancient hydrothermal sites where there are not any geysers or hot springs to signal their existence beneath the surface. Modelling exercises, which can simulate these systems based on what is known and can be observed, are useful in guiding exploration and drilling to tap into these valuable resources.

Part 2 of this document details the steps involved in developing and validating the algorithm to account for the thermal effects of latent heat. Results of simulations provide insight on how the magnitude of the effects of latent heat is dependent on the effects of fluid flow or permeability, the size of the thermal gradient, and the location along the liquid-steam phase curve or the pressure.

Part 3 of this document describes the application of the algorithm developed in Part 2. The algorithm was used to investigate the thermal effects of the latent heat and to interrogate the factors involved in steam formation and distribution within a caldera-hosted hydrothermal system. The conceptual model used was based on the characteristics of several well-documented caldera systems including the Yellowstone caldera in

Yellowstone National Park, USA, the Creede caldera in Colorado, and the Valles Grande caldera in New Mexico. Discussion of the conceptual model's characteristics and the effects of varying several of the system's parameters (e.g., permeability, geometry, intrusion characteristics) are also presented.

REFERENCES

Barnes, H. L. (1997). *Geochemistry of Hydrothermal Ore Deposits*. John Wiley & Sons, New York.

Bowman, J. R., Willett, & Cook, S. J. (1994). Oxygen isotopic transport and exchange during fluid flow: One-dimensional models and applications. *American Journal of Science, 294*, p 1-55.

Bryan, T. S. (2001). *The Geysers of Yellowstone*. 3rd Edition. University Press of Colorado, Boulder, Colorado.

Cook, S. J. & Bowman, J. R. (1994). Contact metamorphism surrounding the Alta stock: Thermal constraints and evidence of advective heat transport from calcite + dolomite geothermometry. *American Mineralogist, 79*, p 513-525.

Cook, S. J. & Bowman, J. R. (2000). Mineralogical evidence for fluid-rock interaction accompanying prograde contact metamorphism of siliceous dolomites: Alta stock aureole, Utah, USA. *Journal of Petrology, 41* (6), p 739-757.

Cook, S. J., Bowman, J. R., & Forster, C. B. (1997). Contact metamorphism surrounding the Alta stock: Finite element model simulation of heat- and $^{18}O/^{16}O$ mass-transport during prograde metamorphism. *American Journal of Science, 297*, p 1-55.

Della Vedova, B., Vecellio, C., Bellani, S., & Tinivella, U. (2007). Thermal modelling of Larderello geothermal field (Tuscany, Italy). *International Journal of Earth Sciences.*

Edwards, R. & Atkinson, K. (1986). *Ore Deposit Geology*. Chapman and Hal, London.

Energy and Geoscience Institute (EGI). (2001). *Geothermal Energy: Clean Sustainable Energy for the Benefit of Humanity and the Environment.* University of Utah.

InfoMine Inc. (2008). *Bingham Canyon.* [Online Resource]. Retrieved October 20th, 2008 from http://www.infomine.com/minesite/.

International Geothermal Association (IGA). (2009*). Italy: Electricity Generation.* [Online Resource]. Retrieved July 30th, 2009 from http://iga.igg.cnr.it/index.php.

Rio Tinto. (2008). Kennecott Utah Copper. [Online Resource]. Retrieved October 20th, 2008 from http://www.kennecott.com/.

Robb, L. (2005). *Introduction to Ore-Forming Processes.* Blackwell Publishing, Malden, MA.

Sass, J. & Priest, S. (2002). Geothermal California. *Geothermal Resource Council (GRC) Bulletin.*

Turcotte, D. L. & Schubert, G. (1982). *Geodynamics: Applications of Continuum Physics to Geological Problems.* John Wiley & Sons, New York.

2. An Approach for Modelling the Effect of Latent Heat on the Liquid-Steam Boundary

2.1. INTRODUCTION

Numerous studies have established the general pattern and magnitude of fluid flow in modern and ancient hydrothermal systems [Bowman, Willett, & Cook, 1994; Cathles, 1977; Cathles et al., 1997; Cook & Bowman, 1994, 2000; Cook, Bowman, & Forster, 1997; EGI, 2001; Giberti, Moreno, & Sartoris, 1984]. The effects of latent heat on the occurrence and distribution of steam in these systems have been largely ignored, and yet the ability to predict the location of the liquid-steam boundary accurately under the conditions prevalent in these hydrothermal systems is essential for determining effective methods of locating and preserving or exploiting these resources.

To address this need for accurate predictions, the primary objective of this study was to develop and test an approach to modelling the effects of latent heat by adding a solution algorithm to a pre-existing numerical model capable of reproducing conditions in hydrothermal systems. The model used was the Aquarius Hydrothermal System Model (Aquarius), a finite-element model developed to simulate fluid flow, advective heat transport, and reactive transport of light stable isotopes. Conditions prevalent in both modern and ancient hydrothermal systems could be simulated including temperatures up to 1000 °C and pressures up to 100 MPa (1 Kb). This system was originally developed reconstruct the hydrothermal flow system responsible for contact metamorphism and ore deposition surrounding the Alta Stock in Utah's Little Cottonwood Mining District [Bowman, Willett, & Cook, 1994; Cook & Bowman, 1994, 2000; Cook, Bowman, & Forster, 1997].

Since Aquarius was originally designed for hydrothermal systems that do not involve a phase change between liquid and steam, the thermal effects of latent heat were not included in the software. Thus, the model was well-suited for testing the approach for modelling the effects of latent heat on the liquid-steam boundary as developed for this study.

Validation of the algorithm and investigation of the effects of latent heat on steam formation were carried out with a series of 2000 m column models made up of 10 000 elements. Model validation is the process of proving that the algorithm or code works correctly [Wang & Anderson, 1982]. Results of simulations provided insight on how the magnitude of the effects of latent heat is dependent on the effects of fluid flow or permeability, the size of the thermal gradient, and the location along the liquid-steam phase curve or the pressure.

2.2 BACKGROUND

2.2.1 Properties of Water

Analysis of fluids collected from active hydrothermal systems and extracted from inclusions in hydrothermally altered rock indicate that they are composed primarily of water, with lesser quantities of carbon dioxide (CO_2), hydrogen sulphide (H_2S), sulphur dioxide (SO_2), and dissolved ionic phases, principally sodium chloride (NaCl) [Barnes, 1997]. Pure water is used as a reasonable first approximation to natural hydrothermal fluids for the purposes of this study since the equations of states for mixed fluids (e.g., H_2O-CO_2-NaCl) are not sufficiently well known to support modelling over a wide range of pressure and temperature conditions [Fontaine & Wilcock, 2007; Hurwitz, Christiansen, & Hsieh, 2007].

The water-dominated nature of typical hydrothermal fluids indicates that properties of water will largely govern the process of hydrothermal convection in natural systems. Figure 2.1 is the portion of the phase diagram of water relevant for studies of hydrothermal systems: the region above the critical point (Pressure: 22.064 MPa; Temperature: 374.1 °C) and the curve between the triple point (Pressure: 0.00061173 MPa; Temperature: 0.01 °C) and the critical point separating the liquid and steam phases [Chang, 2005].

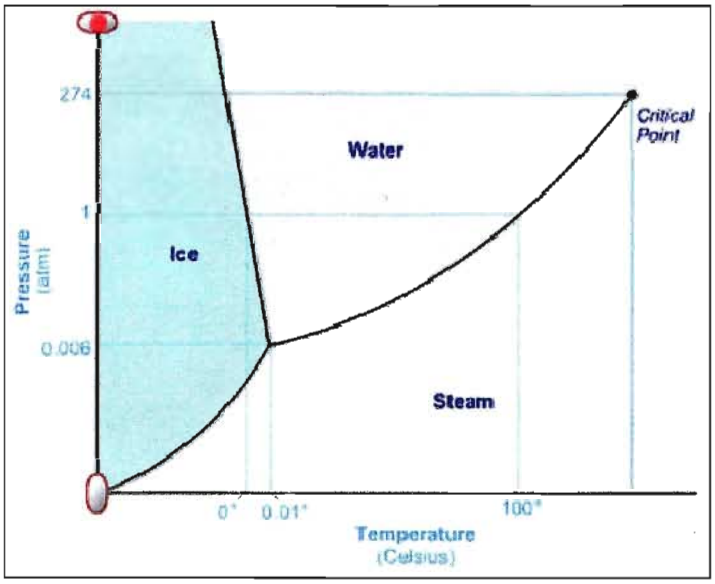


Figure 2.1: The phase diagram of water illustrating pressures and temperatures above the triple point [from Scripps Institute of Oceanography].

Below the critical point, water exists as either a liquid (high density) or vapour (low density) phase. The two phases are separated by the phase equilibrium curve. For pressure and temperature conditions along the curve, energy or latent heat is required for water to change phases. The energy requirements for phase changes as the temperature of water rises under standard pressures conditions (0.1 MPa or 1 bar) are illustrated in Figure 2.2. When the temperature of water reaches the boiling point (100 °C at standard pressure), it will remain at that temperature until a particular amount of energy (2.26×10^6 J/kg for standard pressure) has been added to change all of the liquid water to vapour or steam. This energy is also known as the enthalpy of vaporization. The temperature will begin to rise again once all of the liquid has been changed to steam.

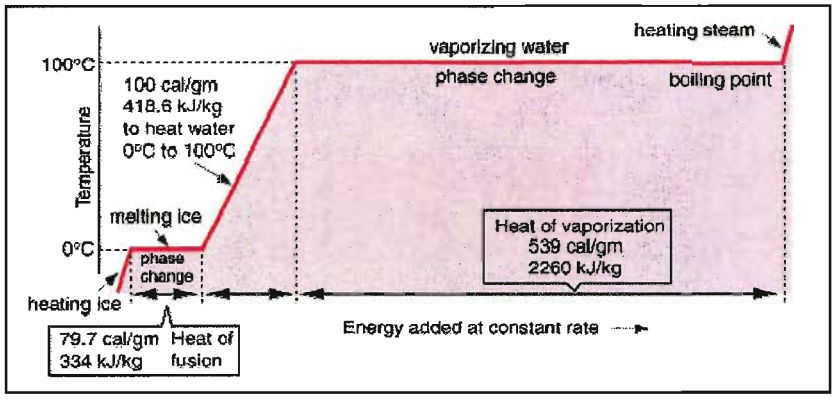


Figure 2.2: Latent heat energy requirements for phase changes of water for standard pressure conditions [Nave, 2005].

As pressures increase from the triple point to the pressure at the critical point (Figure 2.1), the enthalpy of vaporization decreases until reaching zero in the supercritical region. Thus, latent heat will have a greater impact on the thermal state of a system at lower pressures than at pressures close to the critical point. For pressures above the critical point, no phase difference exists, and there is no latent heat of vaporization. These supercritical fluids occur where hydrostatic pressures in the crust exceed the critical point pressure (22.1 MPa).

2.2.2 Convective Hydrothermal Circulation

In hydrothermal systems localized heat sources reduce the density of fluids in surrounding rock. This reduction in density gives rise to buoyant forces that cause less dense fluids to rise. As the fluids rise towards the surface, confining pressures are reduced to below that of the critical point, and the fluids will exist as one of the two phases: liquid or steam. Most of the cooled fluid, then being denser than the heated fluids, will descend once again towards the intrusion, perpetuating the hydrothermal convection [Barnes, 1997; EGI, 2001; Turcotte & Schubert, 1982].

2.2.3 Aquarius Hydrothermal System Model

The Aquarius Hydrothermal System Model (Aquarius) is a finite-element based numerical model developed to model fluid flow, advective heat transport, and reactive transport of light stable isotopes under conditions prevalent in both modern and ancient hydrothermal systems, accommodating temperatures up to 1000 °C and pressures up to 100 MPa (1 Kb). The model has been successfully applied to the reconstruction of the hydrothermal flow system responsible for contact metamorphism and ore deposition surrounding the Alta Stock in Utah's Little Cottonwood Mining District [Bowman, Willett, & Cook, 1994; Cook & Bowman, 1994, 2000; Cook, Bowman, & Forster, 1997].

Originally developed as a set of procedural codes in using C, Aquarius has been reprogrammed in Microsoft C# as an integrated Windows application. The package is equipped with a finite-element mesh generator accommodating left-hand, right-hand, and alternating linear triangles as well as planar horizontal, planar vertical, and cylindrical vertical coordinate sections. Both steady-state and time-transient fluid flow and thermal transport problems can be modelled. Input parameters and solution results can be displayed with the plotting system, and graphics can be printed and saved.

Mathematical Formulation

To model hydrothermal fluid circulation, Aquarius solves two coupled partial differential equations (PDE): one for fluid flow and one for energy transport (temperature). Fluid flow occurs in response to a gradient in fluid potential, which is described in terms of an equivalent freshwater head, h [Bear, 1972; Wang & Anderson, 1982], defined as

$$h = \frac{P}{\rho_o g} + z \qquad \text{(Eq. 2.1)}$$

where P represents fluid pressure, ρ_o a reference fluid density, g the gravitational constant, and z the elevation at which the equivalent head is calculated. The volumetric fluid flux or specific discharge, q_i, through a porous medium, is given by Darcy's law in the following form [Bear, 1972]:

$$q_i = -\frac{k_{ij}\rho_o g}{\mu}\left(\frac{\partial h}{\partial x_i} + \rho_r \frac{\partial z}{\partial x_i}\right) \qquad \text{(Eq. 2.2)}$$

Taken in a specified coordinate direction, i, k_{ij} represents the permeability tensor of the porous medium, μ the kinematic viscosity of the fluid, and ρ_r the relative fluid density, defined as

$$\rho_r = \frac{\rho_f}{\rho_o} - 1 \qquad \text{(Eq. 2.3)}$$

where $\rho_f = \rho_f (p,T)$ represents the actual fluid density. Although conditions in many hydrothermal systems involve fractured media as opposed to porous media, Darcy's law can still be applied as long as the network of fracturing in caldera-hosted hydrothermal settings is dense enough to approximate the behaviour of porous media [Cathles, 1977; Turcotte & Schubert, 1982].

In the absence of fluid sources and sinks, conservation of fluid mass requires the following [Bear, 1972]:

$$\rho_o g(\alpha + \phi\beta)\frac{\partial h}{\partial t} = -\frac{\partial q_i}{\partial x_i} \qquad \text{(Eq. 2.4)}$$

Here α and β represent the compressibilities of the rock matrix and the fluid, respectively, and ϕ is the porosity or pore fraction of the medium. For steady-state conditions, the term $\frac{\partial h}{\partial t} = 0$, and this equation is reduced to be independent of those compressibilities:

$$\frac{\partial q_i}{\partial x_i} = 0 \qquad \text{(Eq. 2.5)}$$

For these calculations, an equivalent freshwater head is used to represent fluid potential for two reasons. First, Eq. 2.4 is stated in terms of the forces driving flow exclusive of static fluid pressure. Second, the use of smaller numbers, the equivalent heads, with relatively large differences is less susceptible to numerical inaccuracies than when calculating gradients using large numbers, the pressures, which have relatively small differences. The fluid pressure can be calculated from equivalent head using Eq. 2.1.

Temperature is governed by the transport equation for energy [Huyakorn and Pinder, 1983]:

$$[\phi\rho_f C_f + (1-\phi)\rho_s C_s]\frac{\partial T}{\partial t} = \frac{\partial}{\partial x_i}\left(\lambda_{ij}\frac{\partial T}{\partial x_j}\right) + \rho_f C_f q_i \frac{\partial T}{\partial x_i} + [\phi\rho_f Q_f + (1-\phi)\rho_s Q_s]$$

(Eq. 2.6)

In this equation T represents the temperature, C_f and C_s the specific heat capacities at constant volume of the fluid and solid, respectively, ρ_s the solid density, Q_f and Q_s the

external rates of heat supply (i.e., heat sources or sinks) associated with the fluid and solid phases respectively, and λ_{ij} the effective conduction-dispersion tensor that models the thermal properties for the fluid and solid. In the above equation, λ_{ij} is a constant value at each step of the calculation determined by the following equations [Huyakorn and Pinder, 1983]:

$$\lambda_{ij} = \phi\lambda_{ij}^{f} + (1-\phi)\lambda_{ij}^{s} \qquad \text{(Eq. 2.7)}$$

where λ_{ij}^{f} represents the conduction-dispersion tensor for the fluid, and λ_{ij}^{s} the thermal conductivity tensor for the solid. For an isotropic media the components of λ_{ij}^{f} are given by [Huyakorn and Pinder, 1983]:

$$\lambda_{ij}^{f} = \rho_f C_f \left(\frac{D_T |q|}{\phi} \delta_{ij} + \frac{(D_L - D_T)}{\phi} \frac{q_i q_j}{|q|} \right) + \lambda_f \qquad \text{(Eq. 2.8)}$$

where $|q|$ represents the magnitude of the fluid flux. D_L and D_T are aquifer parameters that denote the coefficients of longitudinal and transverse dispersivity, respectively. The δ_{ij} is the Dirac Delta function, which is equal to 1 when i and j are the same and 0 otherwise. The λ_f is the thermal conductivity of the fluid. The first term on the right side of Eq. 2.8 is the dispersion component of energy transport. If the water is not moving, this term is eliminated leaving only the diffusion component represented by the second term on the right side of the equation, the thermal conductivity (λ_f).

For steady-state conditions the term $\frac{\partial T}{\partial t}$ from Eq. 2.6 equals 0, and the equation is reduced:

$$\frac{\partial}{\partial x_i}\left(\lambda_{ij}\frac{\partial T}{\partial x_j}\right)+\rho_f C_f q_i \frac{\partial T}{\partial x_i}+\left[\phi\rho_f Q_f+(1-\phi)\rho_s Q_s\right]=0 \qquad \text{(Eq. 2.9)}$$

The first term accounts for conductive-dispersive transport while the second term for advective transport. The third term accounts for energy sources and sinks, and is made up of the addition of two components. The fluid term, Q_f, the first component of energy sources and sinks is nonzero when a phase change occurs. The second component represents the effects of chemical reactions involving the solid phase (i.e., the surrounding rock). For the purpose of this study there is an absence of reactions involving the solid phase. Therefore, $Q_s = 0$, and this second component is ignored.

Fluid and Solid Properties

Aquarius uses established equations and expressions to account for properties of fluids and solids. Fluid densities for temperatures below the critical temperature are calculated with equations from Meyer et al. [1967] and elsewhere equations from Keenan et al. [1978] are used. Other expressions from Keenan et al. [1978] are used to calculate specific heat capacities for fluids while fluid viscosities were calculated from the equations of Watson et al. [1980] and thermal conductivities for fluids from the formulations of Kestin [1978]. Values for enthalpy of vaporization or enthalpy of fusion along the two-phase curve of obtained from the data of Harvey [1995]. Because the specific heat capacities and thermal conductivities for solid phases are relatively insensitive to changes in pressure and temperature, mean values were used as reasonable approximations [Giberti, Moreno, & Sartoris, 1984].

Finite Element Solution Algorithm

The equations governing fluid flow and heat transfer are coupled through the temperature dependence of the fluid properties. Aquarius solves this non-linear problem iteratively. The first step involves calculating the fluid properties for an approximate set of temperature and pressure conditions for each element in the problem domain. Next the equivalent freshwater head values and a fluid flux field are calculated by constructing a global matrix equation using Galerkin's method [Huyakorn & Pinder, 1983] applied to the steady state form of the equation for the conservation of mass, Eq. 2.5. This global equation is then solved using successive over relaxation.

A global matrix equation for the thermal problem is then constructed with Galerkin's method applied to Eq. 2.9 incorporating the calculated the calculated fluid flow field, the model thermal properties, and the existent thermal conditions of the system. This global equation is then solved using successive over relaxation to obtain a new temperature solution. This sequence of steps is repeated until the difference between successive temperature solutions (i.e., the current and previous iterations) is below a given tolerance level (e.g., 0.0001 ^{o}C for column models presented in Section 2.4.1). That is, the system checks for convergence. For each repetition the fluid properties are recalculated using the temperature and pressure conditions that were calculated by the previous iteration.

Galerkin's method is a special case of the method of weighted residuals (MWR) for systems for formulating element matrix equations that together represent the problem domain [Huyakorn & Pinder, 1983]. According to Huyakorn & Pinder [1983] numerical oscillations (overshoot and undershoot) in the Galerkin finite-element solutions can be

nearly eliminated if an element size is selected so that its dimensionless Peclet number is no greater than two. The Peclet number is defined as follows

$$Pe = \frac{V\Delta l}{D} \qquad \text{(Eq. 2.10)}$$

where V is the velocity of flow, Δl is the length of the characteristic length of the finite element, and D is the dispersion coefficient [Bear, 1972; Huyakorn & Pinder, 1983]. Peclet numbers are calculated in Aquarius, and they can be plotted on a diagram of the solution to view the distribution of values.

2.3 EFFECT OF LATENT HEAT ON THE LIQUID-STEAM BOUNDARY

Shallow hydrothermal systems such as the ones at Yellowstone National Park in Wyoming, USA contain a large amount of steam. However, the pressure and temperature conditions accounted in the Alta, Utah hydrothermal system restricted fluids to the supercritical region [Barnes, 1997; Bowman, Willett, & Cook, 1994; Cook & Bowman, 1994, 2000; Cook, Bowman, & Forster, 1997]. Thus, the original Aquarius model code did not need to account for the thermal effects of latent heat on the liquid-steam boundary. To extend the applicability of the model to steam-bearing systems, an approach was needed to account explicitly for the thermal effect of latent heat.

This section presents first a conceptual model of the phase boundary and thermal effects of latent heat at that boundary. Following this qualitative discussion of the problem, the mathematics of the modelling algorithm is detailed. Finally, validation of the modelling algorithm is presented.

2.3.1 Conceptual Model

The phase boundary is visualized as the surface along which fluids existing as liquid and fluids existing as steam are directly adjacent to each other. At this boundary between the two single phases energy is either released or absorbed (i.e., there is an energy source or sink) depending on the direction and rate of flow across the boundary.

Two-phase flow was not modelled in this study for simplicity and consistency with the conceptual models that were used. Since the pressure in the system is held constant (isobaric) due to imposed head values, if flow is up temperature, for example, the temperature of the system cannot change until all of the liquid changes to steam. Trying to overstep this, the steam will only condense faster, while raising the temperature

will only cause the liquid to boil faster. This "buffering" makes it difficult to move along the phase equilibrium curve. To overcome this, the pressure and temperature would need to be adjusted separately at the same time, which is not currently a capability of the Aquarius software.

Further justification for modelling single-phase flow is the rate of flow in the given conceptual model. Low flow rates mean the fluid is barely moving, providing ample time for the system to adjust the phase. If any two-phase flow were to occur given these conditions, it would be a very narrow band. Since each element in the grid can only be in one phase at a time, a narrow band of two-phase flow would appear as a narrow region in which adjacent elements alternate between liquid and steam phases.

The conceptual model accounts for the effects of latent heat on the thermal state of any system in which the phase changes occur between liquid water and steam. The effect of latent heat on the liquid-steam boundary is that of an energy source or sink. Energy is required to change liquid to steam (sink), while energy is released as steam changes to liquid (source). In a steady-state model the effect of incorporating latent heat is to "shift" the liquid-steam boundary from where it is when latent heat is not considered and in the direction of flow. The magnitude of that shift or displacement depends on the magnitude of flow, the thermal gradient, and the position along the two-phase curve.

Element size in the numerical model also affects how far the liquid-steam boundary appears to "shift." Each element in the system can only be one phase at a time meaning that the liquid-steam boundary is always at the element boundary. Thus, in a system where the liquid-steam boundary shifts 20 m, but the elements in the model are 100 m by 100 m, the effects of latent heat might not be observed. Element size is also

important when choosing head and temperature residuals based on head and temperature gradients applied to the system. Values for head and temperature residual should be such that the values are less than the change in head or temperature across the element.

2.3.2 Modelling Algorithm

No alterations were made to the finite element formulations of the equations governing fluid flow or energy transport used by Aquarius. Instead, the latent heat effects arising from the phase change between steam and liquid water were incorporated into the global energy transport equation as point sources and sinks within the global vector that also contains the boundary energy fluxes. In this study, the steady-state form of the energy transport equation (Eq 2.9) was solved to calculate the thermal state of the system.

The process of developing the latent heat algorithm is divisible into five steps:

1. Locate liquid-steam boundary.
2. Calculate the enthalpy.
    a. Locate point between neighbouring element centroids on the liquid-steam boundary.
    b. Determine saturation temperature at point in Step 2a.
    c. Obtain enthalpy value corresponding to the saturation temperature.
3. Determine the component of the fluid flux crossing the liquid-steam boundary.
4. Determine the "direction" of phase change.
5. Calculate the source sink term from fluid density, enthalpy, fluid flux across the liquid-steam boundary, and "direction" of phase change, and divide value to nodes.

Liquid-Steam Boundary

The first step in the process of developing the algorithm to account for the thermal effects of latent heat was to find the liquid-steam boundary. This programming involved examining each steam element in the system and determining if it shared two nodal points (a side) with a liquid element or with the system boundary. Temperature and pressure conditions for each element were determined in the original Aquarius system. Once the elements and nodes along the steam-liquid boundary were identified (Figure 2.3), appropriate enthalpy, fluid flux, and direction values could be determined.

Liquid-Steam Boundary			
Node 1	**Node 2**	**Steam Element**	**Liquid Element**
1	5	1	2
5	9	7	8
9	13	13	14
10	13	15	14
7	10	11	10

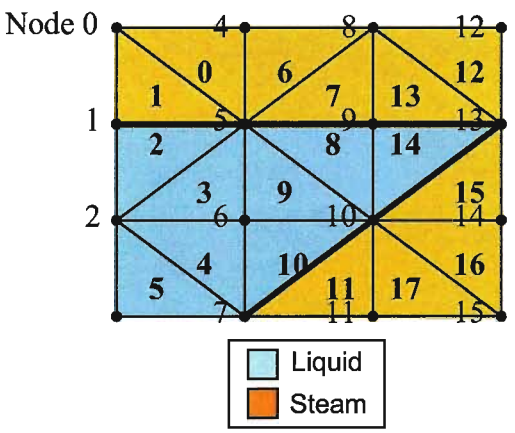


Figure 2.3: Liquid-steam boundary between elements with common nodes. Elements are labelled in bold, while nodes are labelled in regular font style.

Enthalpy

Finding the enthalpy (ΔH), that is, the amount of energy absorbed or released for the phase change to occur, involved 3 steps. First, the point on the boundary where a line between the liquid and steam element centre points or centroids would cross was determined. All possible geometric configurations were considered and outlined in Table 2.1:

Table 2.1: First step in determining enthalpy (ΔH). Rules for locating point between neighbouring element centroids on the liquid-steam boundary.

1. Vertical line between element centroids	
a) Horizontal line between common nodes	The point on the boundary has an x-coordinate equal to that of the element centroids, and a z-coordinate equal to that of the nodal points.
b) Line between common nodes neither horizontal nor vertical	The point on the boundary has an x-coordinate equal to that of the element centroids. The z-coordinate is determined by substituting the x-coordinate into the equation of the line between the nodal points.
2. Horizontal line between element centroids	
a) Vertical line between common nodes	The point on the boundary has an x-coordinate equal to that of the nodal points, and a z-coordinate equal to that of the element centroids.
b) Line between common nodes neither horizontal nor vertical	The point on the boundary has a z-coordinate equal to that of the element centroids. The x-coordinate is determined by substituting the z-coordinate into the equation of the line between the nodal points.
3. Line between element centroids neither horizontal nor vertical	
a) Horizontal line between common nodes	The point on the boundary has a z-coordinate equal to that of the nodal points. The x-coordinate is determined by substituting the z-coordinate into the equation of the line between the element centroids.
b) Vertical line between common nodes	The point on the boundary has an x-coordinate equal to that of the nodal points. The z-coordinate is determined by substituting the x-coordinate into the equation of the line between the element centroids.
c) Line between common nodes neither horizontal nor vertical	The point on the boundary is determined by first finding the slopes and z-intercepts for both the line between the nodal points and the line between the element centroids. The x-coordinate is determined using those slope and intercept values. The z-coordinate is determined by substituting the x-coordinate into either the equation of the line between the nodal points or the equation of the line between the element centroids.

The second step towards finding the enthalpy was to determine the saturation temperature at the point just found on the liquid-steam boundary. Although simple cases were illustrated above, there can be instances when the adjacent elements are not the same size, and therefore, the distances between the element centroids and the boundary

are different. Thus, to interpolate between the liquid and gas element temperatures (T_l *and* T_f, respectively) accurately the temperature of each element and the distances between the liquid element centroid and the boundary point ($dist_l$) and between the gas element centroid and the boundary point ($dist_g$) are used:

$$T_{sat} = T_l + \frac{0 - dist_l}{dist_g - dist_l}(T_g - T_l) \qquad \text{(Eq. 2.11)}$$

The third and final step for determining the enthalpy was to obtain the value corresponding to the saturation temperature from a water property routine based on the data of Harvey [1995] (Appendix 3).

Fluid Flux

In addition to the enthalpy values, the component of fluid flux crossing the liquid-steam boundary was also calculated. That is, the component of the flux that was normal or perpendicular to the boundary was calculated. Again all possible geometric configurations were considered when calculating these values and outlined in Table 2.2:

Table 2.2: Rules for determining the component of fluid flux normal or perpendicular to the liquid-steam boundary (i.e. the component of fluid flux crossing the liquid-steam boundary).

1. Vertical line between element centroids	
a) Horizontal line between common nodes	The normal component of the flux is the z-component of the flux.
b) Line between common nodes neither horizontal nor vertical	The normal component of the flux is determined using a local coordinate system obtained by first finding the angle between the line connecting the nodal points and the x-axis, and then rotating the x component of the flux accordingly.
2. Horizontal line between element centroids	
a) Vertical line between common nodes	The normal component of the flux is the x component of the flux.
b) Line between common nodes neither horizontal nor vertical	The normal component of the flux is determined using a local coordinate system obtained by first finding the angle between the line connecting the nodal points and the x-axis, and then rotating the x component of the flux accordingly.
3. Line between element centroids neither horizontal nor vertical	
a) Horizontal line between common nodes	The normal component of the flux is the z component of the flux.
b) Vertical line between common nodes	The normal component of the flux is the x component of the flux.
c) Line between common nodes neither horizontal nor vertical	The normal component of the flux is determined using a local coordinate system obtained by first finding the angle between the line connecting the nodal points and the x-axis, and then rotating the x component of the flux accordingly.

For instances when neither the line between the element centroids nor the line between the common nodes is horizontal or vertical, a local coordinate system had to be adopted to determine the component of the fluid flux that was normal to the liquid-steam boundary. Shown in Figure 2.4 below is a simple example of this configuration. Figure 2.4 also illustrates the rotation that would need to take place to make the line between the common nodes, treated as a vector, parallel to the x-axis.

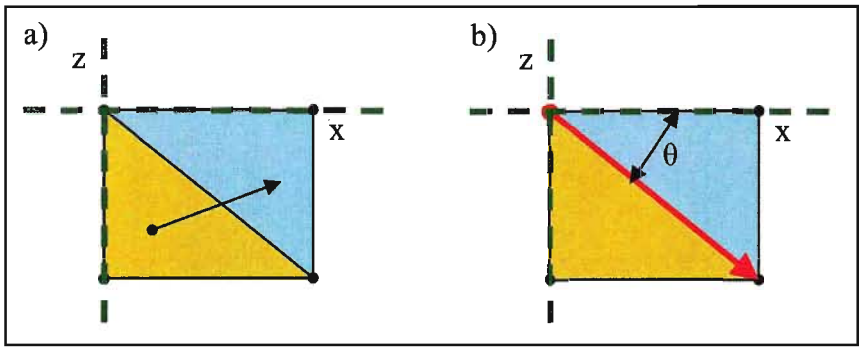


Figure 2.4: Developing a local coordinate system for liquid-steam boundaries that are neither parallel nor perpendicular to the x-axis. **a)** Liquid-steam boundary between two elements and fluid flux vector. b) The liquid-steam boundary between two nodes is considered a vector and divided into its x and z components. The angle, θ, indicates the rotation required for development of the local coordinate system.

The magnitude of this angle was determined by the following where x and z are x- and z-components of the vector between the two nodes:

$$\alpha = \tan^{-1}\left(\frac{x}{z}\right) \qquad \text{(Eq. 2.12)}$$

The system was rotated counter-clockwise according to the following:

Counter-clockwise

$$\begin{bmatrix} x' \\ z' \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ z \end{bmatrix}$$

$$x' = x\cos\theta + z\sin\theta \qquad \text{(Eq. 1.13a)}$$

$$z' = -x\sin\theta + z\cos\theta \qquad \text{(Eq. 1.13b)}$$

If the slope of the liquid-steam boundary was negative (i.e., the first of the two nodes has a greater z coordinate than the second), θ is the angle, α, determined in Eq. 2.12. If the slope of the liquid-steam boundary was positive (i.e. the first of the two nodes has a smaller z coordinate than the second), θ is Pi minus α.

The only value required was the new x-component, x', for the fluid flux value. This value is the component of the flux that is normal or perpendicular to the steam-liquid boundary. Since the direction of flux, for the purposes of the calculation of the energy source/sink term, is dependent on which phase it is flowing from and to, the

magnitude or absolute value of this normal component of flux was taken (i.e. the distance between the new z coordinates of the element centroid and the tip of the flux vector).

Direction of Phase Change

The direction values for the normal component of the flux refer to the direction the fluid is flowing between phases (i.e. steam to liquid or liquid to steam). When the normal component of the fluid flux for a steam element is towards the boundary with the liquid element, the direction is positive, and there is an energy source at this boundary. When the normal component of the fluid flux for a steam element is away from the boundary with the liquid element, the direction is negative, and there is an energy sink at this boundary.

Source/Sink Term

Using the values calculated for enthalpy, the normal component of fluid flux, and the direction of flow with respect to phases of elements, Q_f, the fluid term in the energy transport equation, was determined as follows:

$$Q_f = \frac{q_{\perp} \Delta H}{2} \qquad \text{(Eq. 2.14)}$$

where $q_{\square}$ is the magnitude of the normal component of the flux multiplied by the length of the element boundary and the direction of flow with respect to phase (+1 for steam to liquid, -1 for liquid to steam). The value was divided by two so that it could be added at the nodal points where calculations are made using the energy transport equation.

For systems modelled using radial coordinates, when Q_f is divided out to the common nodes it is multiplied by $2\pi R$, where R is the x-coordinate of the boundary nodes or the distance from the centre of the coordinate system to the node.

2.3.3 Algorithm Validation

Hand calculations, described and demonstrate here, were performed to validate the results from the latent heat algorithm. As the remainder of the Aquarius code was previously validated for point heat sources and sinks, further validation of the original code was unnecessary. Similarly the water properties routines in Aquarius were also validated for previous studies of the reconstruction of the hydrothermal flow system responsible for contact metamorphism and ore deposition surrounding the Alta Stock in Utah's Little Cottonwood Mining District [Bowman, Willett, & Cook, 1994; Cook & Bowman, 1994, 2000; Cook, Bowman, & Forster, 1997]. Since the calculation of the source/sink term as outlined in the previous section was the only piece of Aquarius that needed to be validated for this study.

To aid in algorithm validation, a column model nearly identical to those used in Section 2.5 to test the effects of latent heat on steam distribution. The difference between the column models in Section 1.5 and the validation model here is the size of elements, which were increased for the validation model to illustrate the liquid-steam boundary more clearly. Figure 2.5 illustrates the validation model, which is 2000 m long by 100 m high with elements 20 m wide by 20 m high. Pressure in the system is around 6 MPa. Permeability is 10^{-17} m^2. The temperature gradient is 0.05 °C/m. Table 2.3 lists the relevant raw data used in calculating by hand the source/sink terms for the validation model.

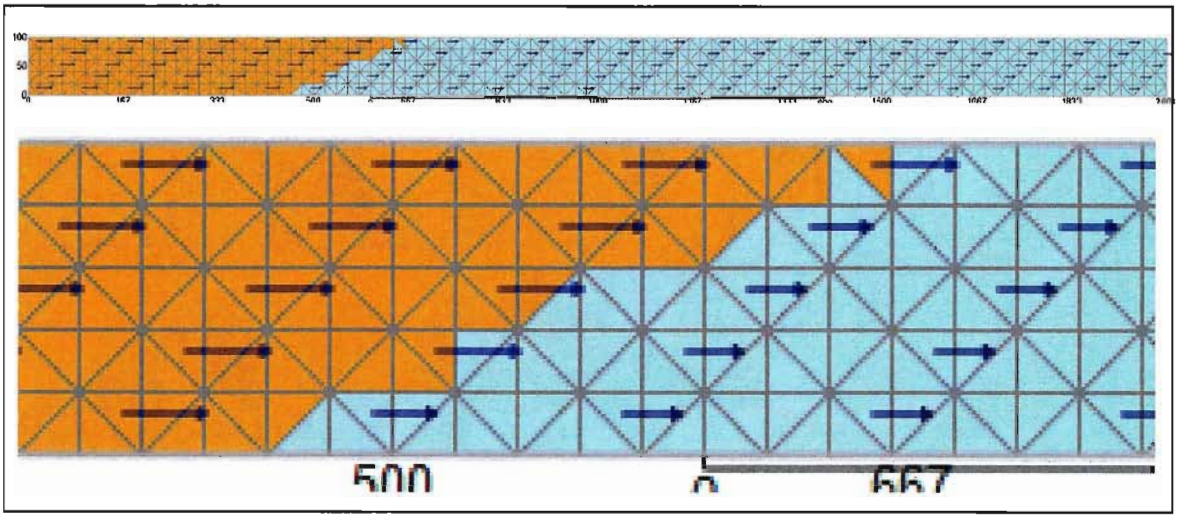


Figure 2.5: Column model used for validation of the latent heat algorithm (top) and close-up view of liquid-steam boundary (bottom).

Table 2.3: Raw data used in calculating by hand the source/sink terms for the validation model.

	Elements		Flux Values					Calculated	
	Steam	Liquid	X (m^2/s)	Z (m^2/s)	Total (m^2/s)	Density (kg/m^3)	Enthalpy (J/kg)	Normal Flux (m^2/s)	Source/ Sink
1	238	239	5.99E-11	5.08E-17	5.99E-11	31.15	1562608	1.2E-09	0.0583
2	247	248	2.07E-11	5.14E-17	2.07E-11	30.40	1569778	-1E-15	-4.9E-08
3	256	266	5.86E-11	5.19E-17	5.86E-11	30.07	1577912	1.17E-09	0.0556
4	257	258	2.07E-11	5.15E-17	2.07E-11	30.50	1573873	-1E-15	-4.9E-08
5	265	266	1.97E-11	5.11E-17	1.97E-11	29.22	1579936	-1E-15	-4.7E-08
6	274	275	5.72E-11	5.17E-17	5.72E-11	28.89	1587021	1.14E-09	0.0524
7	283	284	1.88E-11	5.16E-17	1.88E-11	28.15	1594018	-1E-15	-4.6E-08
8	293	294	1.88E-11	5.16E-17	1.88E-11	28.24	1598016	-1E-15	-4.7E-08
9	302	303	5.46E-11	5.21E-17	5.46E-11	27.90	1604907	1.09E-09	0.0489
10	311	312	1.76E-11	5.22E-17	1.76E-11	27.15	1611817	-1E-15	-4.6E-08
11	311	321	1.76E-11	5.22E-17	1.76E-11	27.15	1613794	3.52E-10	0.0154
12	320	321	5.31E-11	5.28E-17	5.31E-11	26.82	1616743	-1.1E-09	-0.0460
13	320	330	5.31E-11	5.28E-17	5.31E-11	26.82	1619648	1.06E-09	0.0461

The calculations shown here for this model are representative of the calculations performed to validate the ability of the latent heat algorithm to account for all element geometries and liquid/steam arrangements. To validate that the latent heat algorithm has correctly identified the liquid-steam boundary, the element and node numbers listed in Table 2.3 are compared with the elements and nodes along the liquid-steam boundary in

Figure 2.5 by visual inspection. The direction of phase change represented by the sign of the normal flux in Table 2.3 can also be validated by visual inspection, remembering that when the normal component of the fluid flux for a steam element is towards the boundary with the liquid element, the direction is positive (source), and when the normal component of the fluid flux for a steam element is away from the boundary with the liquid element, the direction is negative (sink).

When the liquid-steam boundary is horizontal (Table 2.3 lines 2, 4, 5, 8, and 10), the normal component of the flux is the z-component of the flux. When the liquid-steam boundary is vertical (Table 2.3 lines 3, 11, and 13), the normal component of the flux is the x-component of the flux. The magnitude of the normal component of the flux is multiplied by the density, the length of the element boundary, and the direction of flow with respect to phase (+1 for steam to liquid, -1 for liquid to steam). The value was then multiplied by the enthalpy of vaporization to give the source/sink term.

For example, the calculations for Table 2.3 lines 2 and 3 are as follows:

Line 2: Horizontal Boundary

$$q_{\perp} = \left(5.14\times10^{-17}\right)(30.40)(20)(-1) = -3.12512\times10^{-14}$$

$$Q_f = \left(-3.12512\times10^{-14}\right)(1569778) = -4.905\times10^{-08}$$

Line 3: Vertical Boundary

$$q_{\perp} = \left(5.86\times10^{-11}\right)(30.07)(20)(1) = 3.52420\times10^{-08}$$

$$Q_f = \left(3.52420\times10^{-08}\right)(1577912) = 0.05561$$

When the liquid-steam boundary is neither horizontal nor vertical (Table 2.3 lines 1, 6, 9, and 12), the component of the flux normal to the boundary is calculated by rotating the coordinate system as described in Section 2.3.2. A set of tables for

calculating the normal component of the flux and the source/sink value is presented in Appendix 4 and allows for all geometries and liquid/steam arrangements when the liquid-steam boundary is neither horizontal nor vertical. Lengths for each side of the element can be entered or calculated. The angle of rotation is calculated using Eq. 2.12. Eq. 2.13 is applied according to the slope of the liquid-steam boundary. The normal component of the flux and the source/sink term are calculated as described above with values being multiplied by -1 when the liquid element is above the steam element.

For example, the calculations for Table 2.3 lines 1 (positive slope), 6 (positive slope), 9 (positive slope), and 12 (negative slope), all of which had steam elements above liquid elements, are as follows:

	Line 1	Line 6	Line 9	Line 12
Length	28.28	28.28	28.28	28.28
θ	0.7854	0.7854	0.7854	0.7854
$\pi - \theta$	2.3562	2.3562	2.3562	N/A
$q_{\square}$	1.1984E-09	1.1430E-09	1.0929E-09	-1.0613E-09
Q_f	0.058339	0.05241	0.04894	-0.04603

Secondary Validation: Mass Balance

A mass balance calculation for the column models shows that the mass flux fields across the inflow and outflow boundaries match. Fluid flux and density values for boundary elements from the model were used as the raw data for calculating the mass flux into the model and the mass flux out of the model:

In			Out		
Fluid Flux (m^2/s)	Density (kg/m^3)	Mass Flux (kg/s)	Fluid Flux (m^2/s)	Density (kg/m^3)	Mass Flux (kg/s)
1.212E-09	2.4438E+01	2.9619E-08	3.580E-11	8.6719E+02	3.1045E-08
1.212E-09	2.5065E+01	3.0379E-08	3.580E-11	8.6728E+02	3.1049E-08
1.286E-09	2.6551E+01	3.4145E-08	4.140E-11	8.6748E+02	3.5914E-08
1.286E-09	2.7311E+01	3.5122E-08	4.140E-11	8.6758E+02	3.5918E-08
1.332E-09	2.8865E+01	3.8448E-08	4.520E-11	8.6778E+02	3.9178E-08
	Total:	1.67713E-07		Total:	1.73104E-07

Since fluid flux values were given at element centres or centroids, those values were multiplied by the length of the side of the element at the boundary, 20 m. Also, because flux values are calculated at element centroids, they represent only approximate flux values for the element boundaries. This is why the difference between the mass flux out and the mass flux in is not exactly zero. With a difference in mass flux of only 5.391×10^{-09} kg/s, this mass balance validates that mass is conserved.

2.4 MODELS AND RESULTS

2.4.1 Column Models

A series of column models were constructed to evaluate the latent heat algorithm and to examine the effect of latent heat on the location of the liquid-steam boundary under natural (i.e., field scale) conditions. Columns were 2000 m long and 100 m high and divided into 10 000 elements, 2 m wide by 20 m high.

The models were designed such that head and temperature conditions were fixed at either end of the column, forcing flow in one direction and producing a single liquid-steam boundary, as illustrated in Figure 2.6. Head gradients were selected to maintain nearly isobaric conditions at approximately 6 MPa, 12 MPa, and 18 MPa. For each pressure, three temperature gradients were simulated: 0.05 °C/m, 0.10 °C/m, and 0.15 °C/m. For each of the pressure-temperature gradient combinations, five permeabilities were modelled (10^{-18} m^2, 10^{-17} m^2, 10^{-16} m^2, 10^{-15} m^2, and 10^{-14} m^2 in the x direction; 10^{-25} m^2 in the z-direction) to simulate different flow rates.

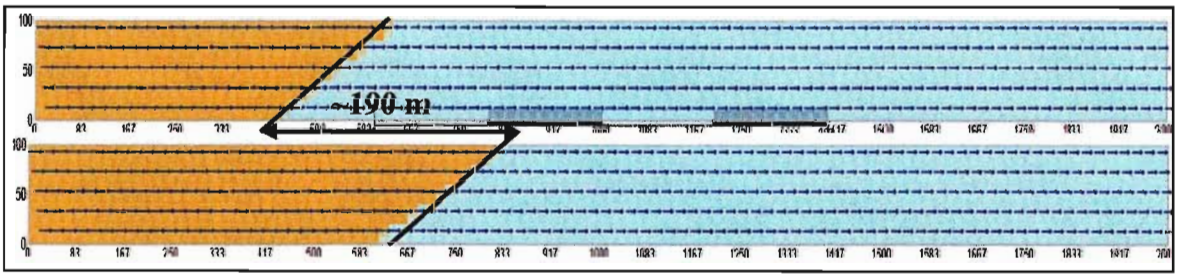


Figure 2.6: Comparison of column models run without effects of latent heat (top) and with effects of latent heat (bottom). Flow Direction: Down temperature. Pressure: 6 MPa. Thermal Gradient: 0.05 °C/m Permeability: 10^{-16} m^2. Magnitude of Displacement: 190 m

Each scenario was run twice: once without the latent heat algorithm and once with the latent heat algorithm. The location of the liquid-steam boundary in the column was noted and compared for each, yielding a displacement value. This value was used to

quantify the effects of the latent heat or the magnitude of the "shift" or displacement of the boundary. Displacement values are listed in Tables A5.1 and A5.2 in Appendix 5.

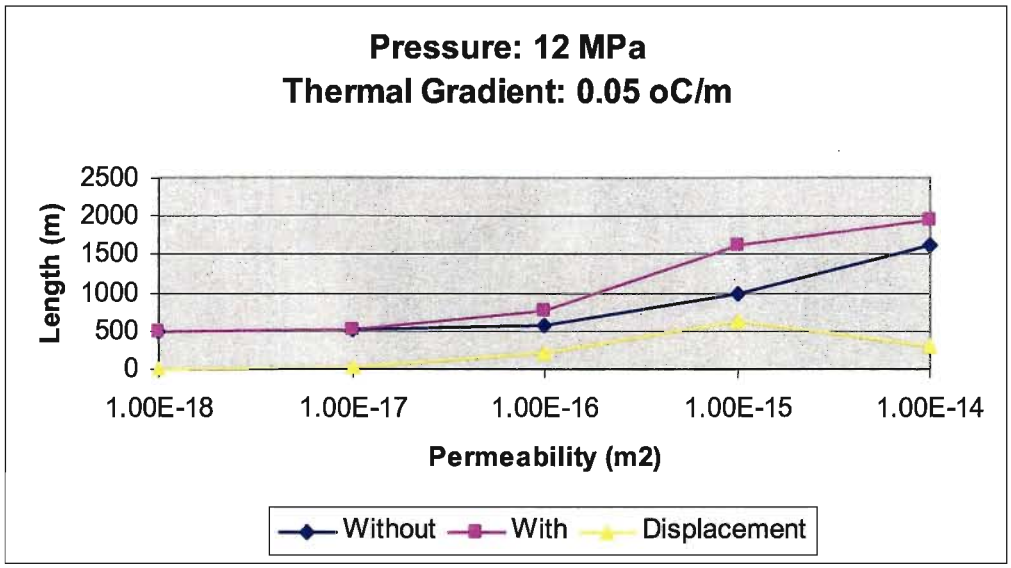
Solution residuals for hydraulic head were 0.0001 m and for temperature were 0.0001 °C to ensure the accuracy of the solution given the small head gradients (10 m/2000 m) and temperature gradients (0.05 °C/m, 0.10 °C/m, and 0.15 °C/m) of the systems.

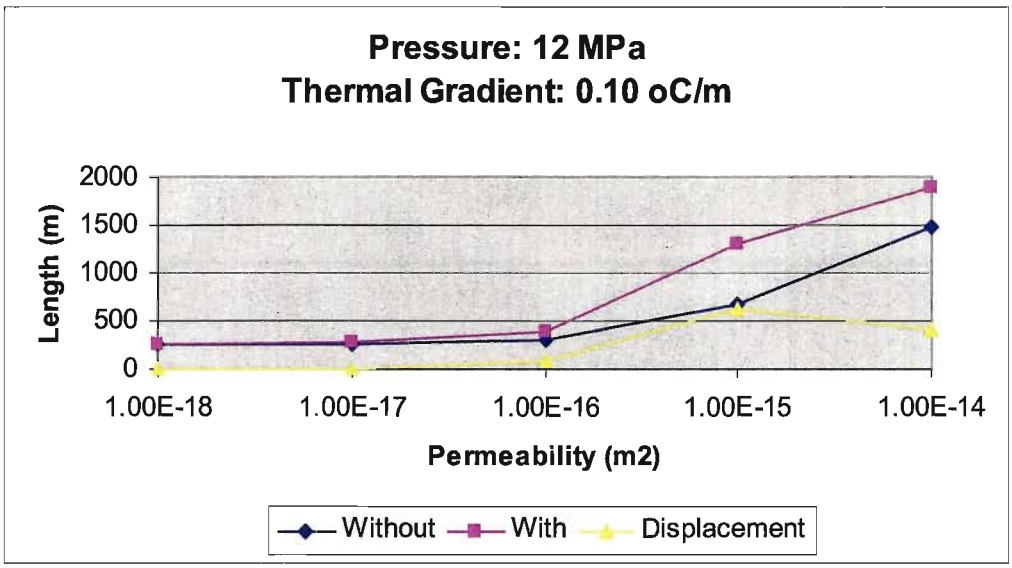
2.4.2 Effect of Flow Rate (Permeability)

For models run with a permeability of 10^{-18} m^2 to simulate conductive materials, there was no displacement of the liquid-steam boundary when the effects of latent heat were included. This was the case for both up and down temperature flow directions, for each of thermal gradient, and for each position along the two-phase curve (pressure). Models with a permeability of 10^{-17} m^2 allowed for some convection and typically exhibited a shift in the liquid-steam boundary when the effects of latent heat were included. The magnitude of this shift ranged from approximately 2.5 m to 23.0 m with the lower displacement values observed with higher pressures and larger thermal gradients and the higher values with lower pressures and smaller temperature gradients.

For models run with permeability of 10^{-16} m^2, displacement values ranged from approximately 15.0 m to 210.0 m, again with the lower displacement values being observed with higher pressures and larger thermal gradients and the higher values with lower pressures and smaller temperature gradients. Sections 2.4.3 and 2.4.4 will further describe the effects of the thermal gradient and the position along the two-phase curve on the displacement values. Displacement values ranged from approximately 17 m to 1322 m for models with permeability of 10^{-15} m^2 and from approximately 33 m to 630 m for

models with permeability of 10^{-14} m^2. This apparent reversal in displacement value ranges for the highest two permeabilities is the result of the effects of the system or column boundary. Figure 2.7 illustrates this trend with each of the three temperature gradients for a pressure of 12 MPa.





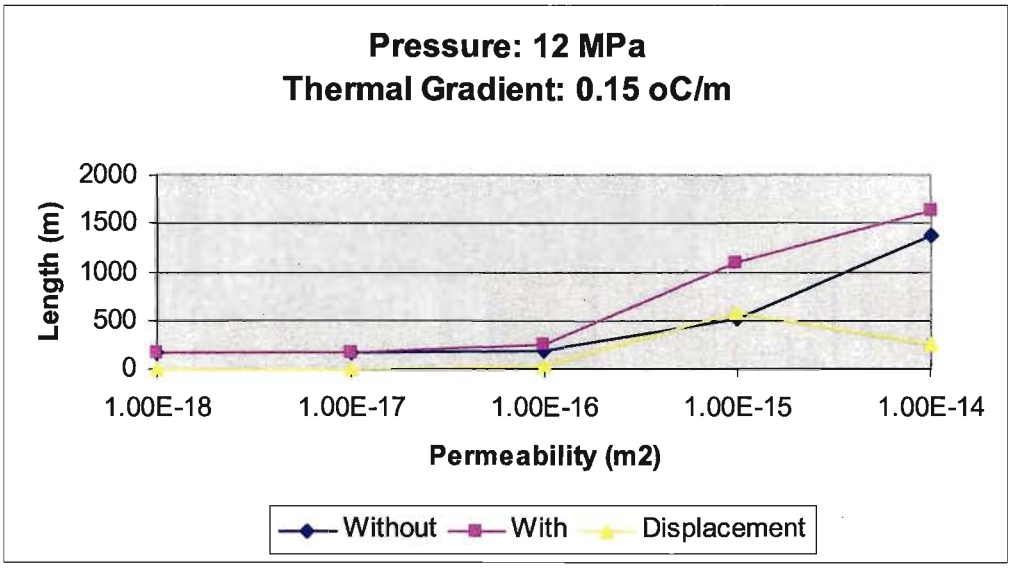


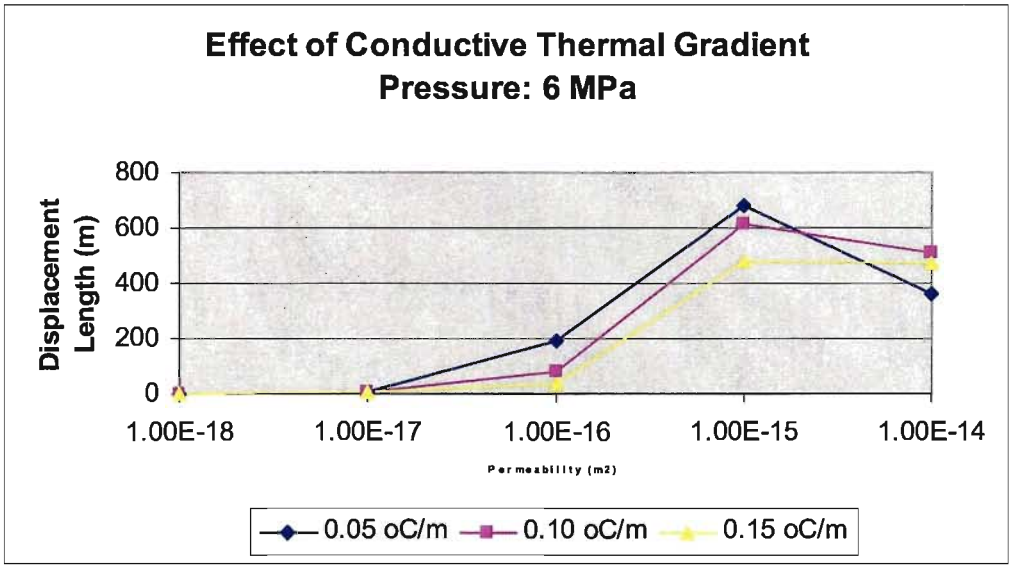
Figure 2.7: Effect of flow rate (permeability) on the magnitude of displacement for down temperature flow. Distances without and with the effects of latent heat as well as the difference between the two are plotted against the range of five permeabilities (10^{-18} to 10^{-14} m^2) for each of the three temperature gradients.

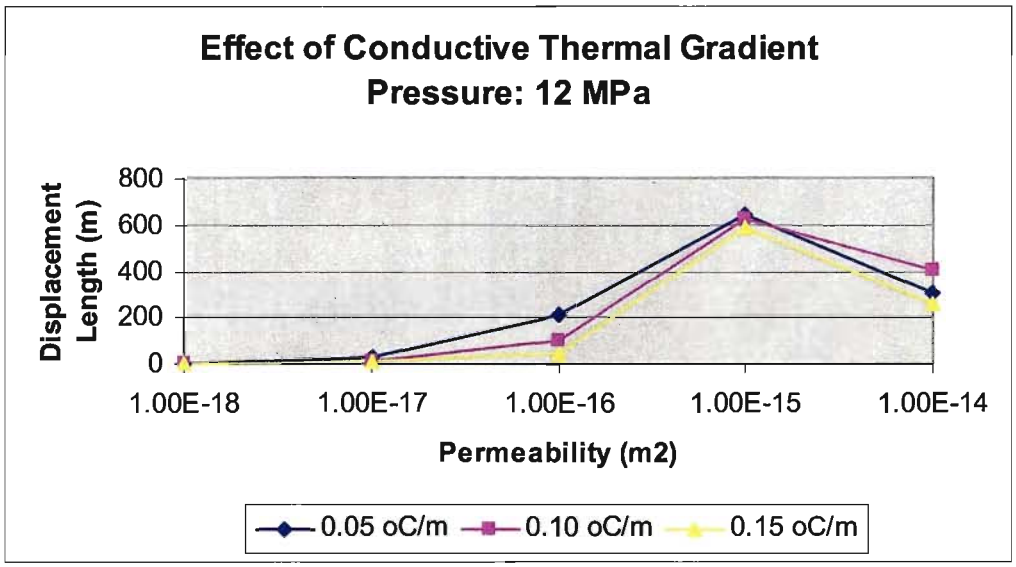
2.4.3 Effect of Conductive Thermal Gradient

Three temperature gradients were simulated at each of the three pressures: 0.05 °C/m, 0.10 °C/m, and 0.15 °C/m. As the gradient gets larger, the effect of latent heat becomes smaller. For example, moving down temperature or from steam to liquid, more energy is required to raise the boiling point of the liquid. With larger temperature gradients, the liquid is cool enough to absorb all of the energy released from the phase change, and still not reach the boiling point. Thus, as the temperature gradient increases, the magnitude of the "shift" of the boundary becomes smaller. Figure 2.8 illustrates this trend for the three thermal gradients (0.05, 0.10, and 0.15 °C/m) plotted against the range of five permeabilities (10^{-18} to 10^{-14} m^2) for each of the three pressures.

At the higher permeabilities (greater than 10^{-16} m^2) or the higher flow rates that were simulated, the lines representing the three thermal gradients are shown to cross. Again, this reversal occurs when the effects of the system boundary are greater than the

effects of the latent heat. If an infinite or a semi-infinite column were used all three of the lines would have continued to increase with increasing permeability.





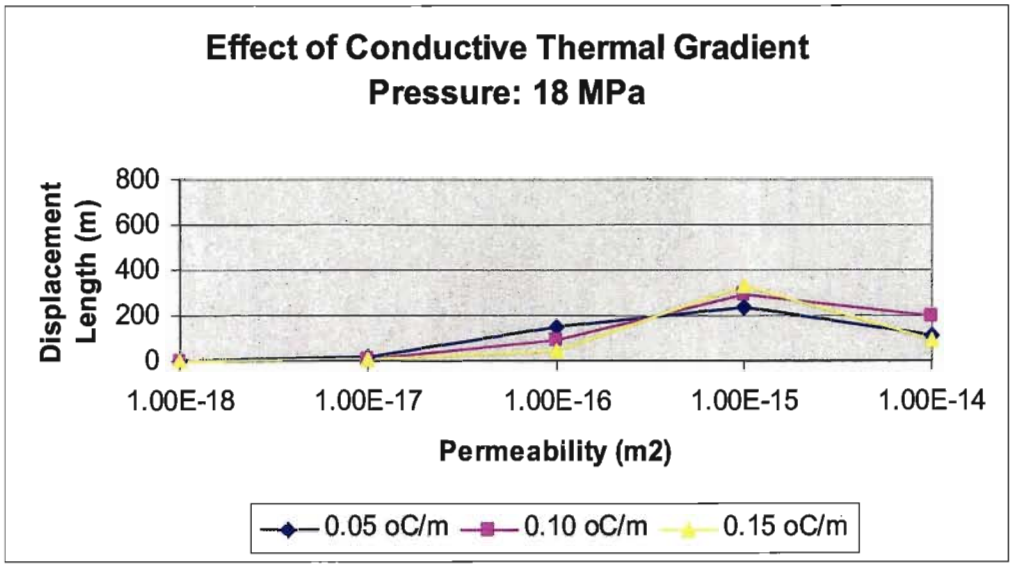
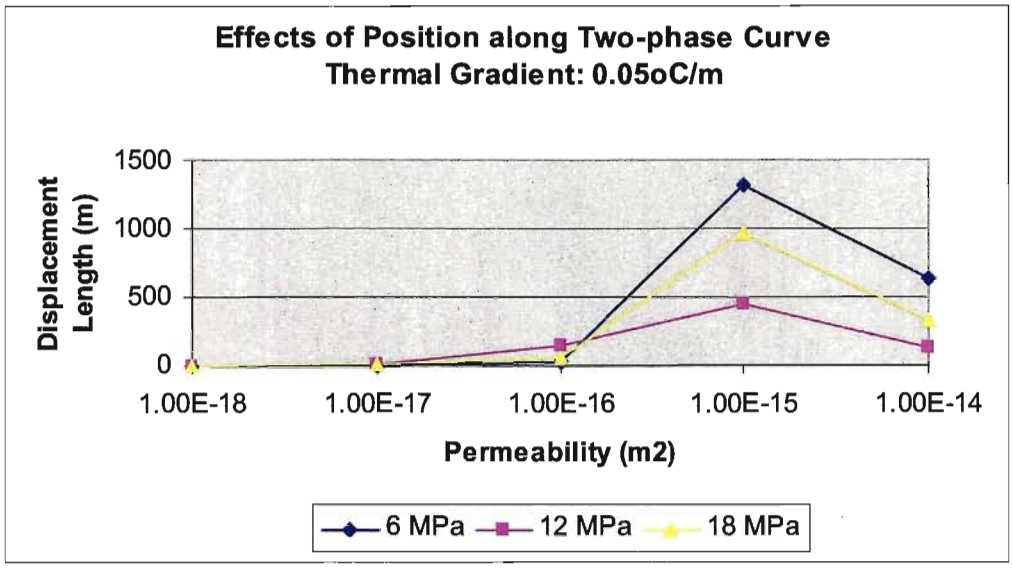
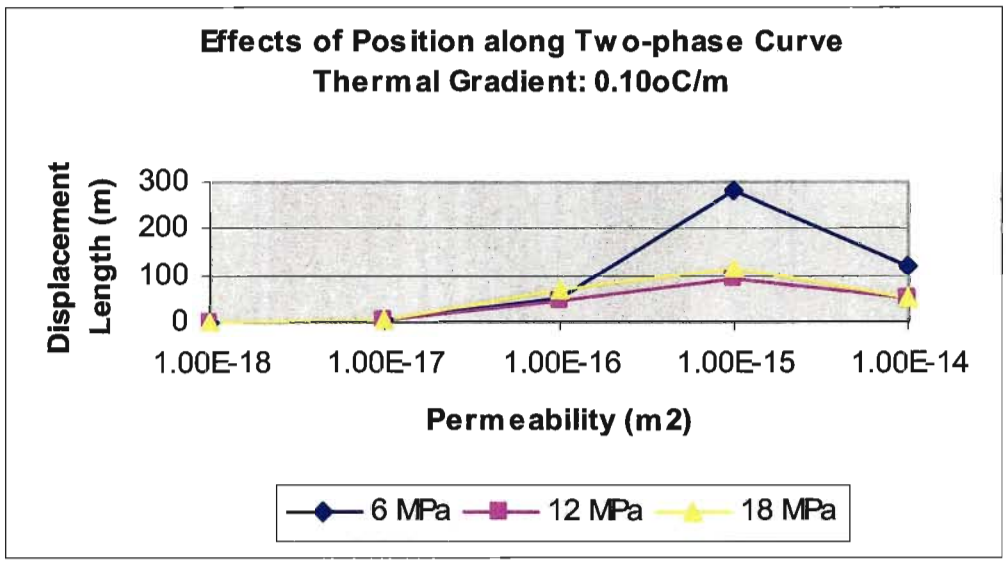


Figure 2.8: Effect of conductive thermal gradient on the magnitude of displacement for down temperature flow. Displacement values for the three thermal gradients (0.05, 0.10, and 0.15 °C/m) are plotted against the range of five permeabilities (10^{-18} to 10^{-14} m^2) for each of the three pressures.

2.4.4 Effect of Position along Two-Phase Curve

Along the phase equilibrium curve separating the liquid and steam phases of water, the amount of latent heat energy required for a phase change decreases to zero at the critical point (T = 373.946 °C, P = 22.064 MPa) beyond which water exists as a supercritical fluid. Thus, in comparing the effects of latent heat at each of the three pressures (6 MPa, 12 MPa, and 18 MPa; Figure 2.9), it is evident that the effects of latent are reduced as pressure is increased. The effects of the system or column boundary are evident at higher permeabilities when the liquid-steam boundary approaches the system boundary.


Effects of Position along Two-phase Curve
Thermal Gradient: 0.05oC/m
Displacement Length (m)
1500
1000
500
0
1.00E-18
1.00E-17
1.00E-16
1.00E-15
1.00E-14
Permeability (m2)
6 MPa
12 MPa
18 MPa


Effects of Position along Two-phase Curve
Thermal Gradient: 0.10oC/m
Displacement Length (m)
300
200
100
0
1.00E-18
1.00E-17
1.00E-16
1.00E-15
1.00E-14
Permeability (m2)
6 MPa
12 MPa
18 MPa

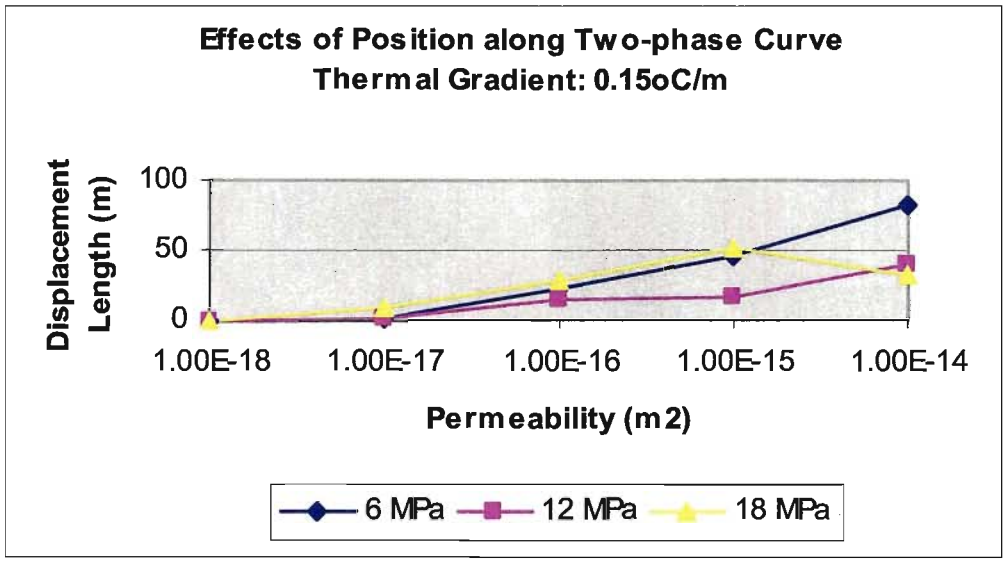


Figure 2.9: Effect of position along the two-phase curve on the magnitude of displacement for down temperature flow. Displacement values for the three pressures (6, 12, and 18 MPa) are plotted against the range of five permeabilities (10^{-18} to 10^{-14} m^2) for each of the three thermal gradients.

2.5 DISCUSSION

2.5.1 Effect of Column Boundary

The flow rate, the thermal gradient, and the position along the two-phase curve are not the only parameters and conditions that impact the effects of latent. Reviewing Figures 2.7, 2.8, and 2.9, it can be observed that as the liquid-steam boundary approaches the end of the column, the magnitude of the boundary "shift" due to the effects of latent heat is reduced. In an infinite or a semi-infinite column, this reduction would not occur. Because these column models are finite in length (2000 m) and head and temperature values have been fixed at both ends, the system or column boundary affects the amount that the liquid-steam boundary can be displaced. With the fixed conditions the liquid-steam boundary cannot be forced beyond the end of the column. As seen in Figure 2.10, this does push the temperature contours closer together at the end of the column creating a steeper gradient because of the system boundary.

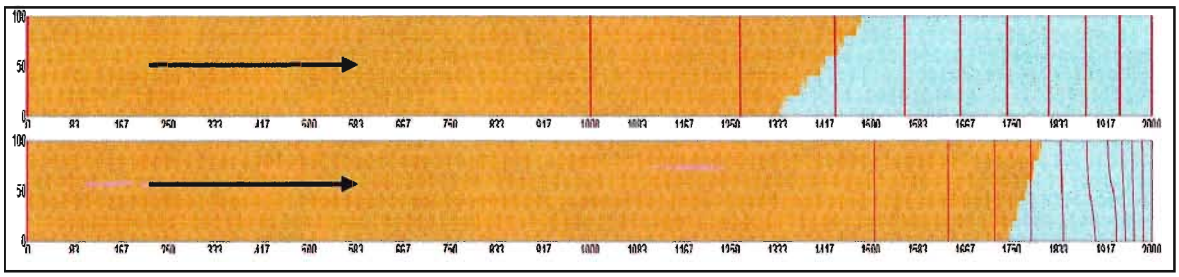


Figure 2.10: Comparison of temperature contours (10 °C intervals) in column models run without effects of latent heat (top) and with effects of latent heat (bottom). Flow Direction: Down temperature. Pressure: 6 MPa. Thermal Gradient: 0.05 °C./m Permeability: 10^{-14} m^2.

2.5.2 Liquid-Steam Interface

The liquid-steam boundary is not a straight line from the bottom to the top of the column as a result of variations in the properties of the liquid and steam along that boundary. These variations relate to the pressure in the system, which increases with

depth, and to the density differences between liquid water (high density) and steam (low density). Figure 2.11 illustrates the liquid-steam interface.

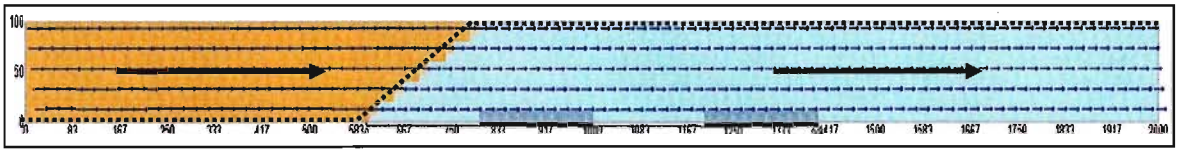


Figure 2.11: Liquid-steam boundary is not a straight line due to variations in the properties of liquid and steam.

2.5.3 Two-Phase Flow

Along the liquid-steam boundary in the column model, a narrow region where the phase alternates between liquid and steam occurs (Figure 2.12). This alternating pattern, which is approximately 2 to 5 elements (4 to 10 m) wide, might indicate a region where two-phase flow would occur. Given that each element in the model can only be one phase at a time, each element is forced to either liquid or steam, when in reality there might be both liquid and steam flowing together in that narrow band where the phase change occurs.

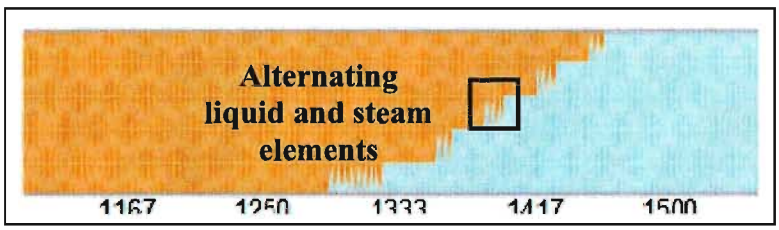


Figure 2.12: Narrow region where phase alternates between liquid and steam along the liquid-steam boundary. This indicates the possibility of two-phase flow in this region.

2.6 SUMMARY AND CONCLUSIONS

To investigate the thermal effects associated with vaporization evident in ancient and modern hydrothermal settings, a procedure was developed to account for latent heat effects and added to an existing software package, the Aquarius Hydrothermal System Model. That algorithm incorporated the liquid-steam phase change into the energy transport equation modelled by Aquarius.

A series of essentially one-dimensional column models were constructed to validate the latent heat algorithm and to investigate the effects of certain parameters and conditions on the magnitude of the effects of latent heat. The effects of flow rate, thermal gradient, and position along the two-phase curve were investigated and the following trends were observed:

- As the flow rate (permeability) is increased, the magnitude of displacement of the liquid-steam boundary from an initial position determined without accounting for latent heat is increased until the effects of the system boundary were greater than the effects of the latent heat.
- As the thermal gradient is increased, the magnitude of displacement is reduced until the effects of the system boundary were greater than the effects of the latent heat. In flowing from steam to liquid, for example, the larger difference in temperature means that the liquid is cool enough to absorb all of the energy released from the phase change without reaching the boiling point.
- As pressure is increased or the position along the two-phase curve approaches the critical point, the magnitude of displacement is reduced due to decreasing values

of latent heat energy until the effects of the system boundary were greater than the effects of the latent heat.

It was observed that the column length or system boundary acts to reduce the magnitude of displacement due to the conditions imposed or fixed at that boundary. The liquid-steam boundary is not a straight line from the top to the bottom of the column due to variations in the properties of liquid and steam due to pressure and density conditions. A narrow region at the liquid-steam boundary indicates that two-phase flow would likely occur over a narrow range (4 to 10 m). This was manifested as a region of alternating liquid and steam elements since each element in the grid might only be in one phase at a time in the Aquarius system.

2.7 REFERENCES

Barnes, H. L. (1997). *Geochemistry of Hydrothermal Ore Deposits*. John Wiley & Sons, New York.

Bear, J. (1972). *Dynamics of Fluids in Porous Media*. Dover Publications, Inc., Mineola, New York.

Bowman, J. R., Willett, & Cook, S. J. (1994). Oxygen isotopic transport and exchange during fluid flow: One-dimensional models and applications. *American Journal of Science, 294*, p 1-55.

Bryan, T. S. (2001). *The Geysers of Yellowstone*. 3rd Edition. University Press of Colorado, Boulder, Colorado.

Cathles, L. M. (1977). An analysis of the cooling of intrusives by ground-water convection which includes boiling. *Economic Geology, 72*, p 804-826.

Cathles, L. M., Erendi, A. H. J., & Barrie, T. (1997). How long can a hydrothermal system be sustained by a single intrusive event? *Economic Geology, 92,* p 766-771.

Cook, S. J. & Bowman, J. R. (1994). Contact metamorphism surrounding the Alta stock: Thermal constraints and evidence of advective heat transport from calcite + dolomite geothermometry. *American Mineralogist, 79*, p 513-525.

Cook, S. J. & Bowman, J. R. (2000). Mineralogical evidence for fluid-rock interaction accompanying prograde contact metamorphism of siliceous dolomites: Alta stock aureole, Utah, USA. *Journal of Petrology, 41* (6), p 739-757.

Cook, S. J., Bowman, J. R., & Forster, C. B. (1997). Contact metamorphism surrounding the Alta stock: Finite element model simulation of heat- and $^{18}O/^{16}O$

mass-transport during prograde metamorphism. *American Journal of Science, 297*, p 1-55.

Energy and Geoscience Institute (EGI). (2001). *Geothermal Energy: Clean Sustainable Energy for the Benefit of Humanity and the Environment.* University of Utah.

Fontaine, F. J. & Wilcock, W. S. D. (2007). Two-dimensional numerical models of open-top hydrothermal convection at high Rayleigh and Nusselt numbers: Implications for mid-ocean ridge hydrothermal circulation. *Geochemistry, Geophysics, Geosystems, 8*, No. 7.

Freeze, R. A. & Cherry, J. A. (1979). *Groundwater.* Prentice Hall, Inc., Englewood Cliffs, New Jersey.

Giberti, G., Moreno, S., & Sartoris, G. (1984). Evaluation of approximations in modelling the cooling of magmatic bodies. *Journal of Volcanology and Geothermal Research, 20*, p 297-310.

Gourley, B., Finley, R., & Gourley, T. (1997-2005). *Introduction to Hydrothermal (Steam) Explosions in Yellowstone.* [Online Resource]. Retrieved December 4th, 2007 from http://www.yellowstone.net/hydrothermal.htm.

Harvey, A. H. (1995). *Thermodynamic Properties of Water: Tabulation from the IAPWS Formulation 1995 for the Thermodynamic Properties of Ordinary Water Substance for General and Scientific Use.* National Institute of Standards and Technology, Boulder, Colorado.

Hurwitz, S., Christiansen, L. B, & Hsieh, P. A. (2007). Hydrothermal fluid flow and deformation in large calderas: Inferences from numerical simulations. *Journal of Geophysical Research, 112.*

Huyakorn, P. S. & Pinder, G. F. (1983). *Computational Methods in Subsurface Flow.* Academic Press, Inc., Orlando, Florida.

Keenan, J. H., Keyes, F. G., Hill, P.G., & Moore, J. G. (1978). *Steam Tables.* John Wiley, New York.

Kestin, J. (1978). Thermal conductivity of water and steam. *Mechanical Engineering, 100,* p 1255-1258.

Meyer, C. A., McClintoch, R. B., Silvestri, G. J., & Spencer, R. C. (1967). ASME steam tables – Thermodynamic and transport properties of steam. *American Society of Mechanical Engineers.*

Scripps Institution of Oceanography. (2009). *Earth Guide Online Classroom: Water Phase Diagram.* [Online Resource]. Retrieved January 4th, 2009 from http://earthguide.ucsd.edu/

Turcotte, D. L. & Schubert, G. (1982). *Geodynamics: Applications of Continuum Physics to Geological Problems.* John Wiley & Sons, New York.

Wang, H. F. & Anderson, M. P. (1982). *Introduction to Groundwater Modelling: Finite Difference and Finite Element Methods.* W. H. Freeman and Company, San Francisco.

3. Controls on the Occurrence and Distribution of Steam in Caldera-Hosted Hydrothermal Systems

3. 1 INTRODUCTION

The formation of steam is a fundamental process in shallow hydrothermal systems, both ancient and modern. While ancient hydrothermal systems show evidence of boiling, a direct examination of the spatial and temporal evolution of those systems is not possible. Only the results of boiling are preserved. Even in modern or active systems such as the Yellowstone system, direct observations of the hydrothermal system are difficult to obtain. The pressure in the system and the corrosive characteristics of hydrothermal solutions both pose technological challenges, and the installation of observation wells alters the existing conditions. Models that can simulate conditions in these systems, when constrained by observations of known systems and the characteristics of their host rocks, can provide additional insights into the thermal evolution of ancient systems and help to guide exploration of active systems and aid in the preservation of their unique features.

The secondary objective of this study was to examine the impact of including the effects of latent heat on the occurrence and distribution of steam under environmental conditions found in hydrothermal systems. The objective was met by applying the modelling approach detailed in Part 2 to an idealized caldera-hosted hydrothermal system based on a conceptual model derived primarily from the characteristics of several well-documented caldera systems: the Yellowstone caldera in Wyomin, USA, the Creede caldera in Colorado, USA, and the Valles Grande caldera in New Mexico, USA.

In this investigation of a regional-scale hydrothermal system, a sensitivity analysis was performed to interrogate additional factors (e.g., intrusion characteristics,

permeability, surface topography) that could potentially influence steam formation and distribution. This analysis did not include simulations with the latent heat algorithm due to issues of convergence and model stability (See Section 3.4.1).

3.2 CHARACTERISTICS OF CALDERA-HOSTED HYDROTHERMAL SYSTEMS

In this study, hydrothermal activity was modelled for an idealized caldera setting. The characteristics of this caldera were selected from the parameters representative of a suite of modern and ancient calderas including the Yellowstone, Creede, and Valles Grande systems.

Although a significantly larger, more complex system than the model designed for this study, the 45 km by 85 km Yellowstone system with two resurgent domes offers evidence of near-surface boiling or steam generation observable in the thousands of thermal features visible in the park [Bryan, 2001; Christiansen, 2001]. Further studies of this specific site, a designated national park, could enhance the preservation techniques for the hydrothermal system.

The Creede caldera is the best preserved caldera in the San Juan caldera cluster, which also includes Bachelor, La Garita, and San Luis calderas. Creede is significant to this study not only because it contains epithermal vein deposits of base- and precious-metals, but also because it is a resurgent caldera [Lipman, 1997, 2006]. The caldera floor can "resurge" or bulge due to subsequent intrusion of magma. Sections 3.3 and 3.4 outline how various caldera floor topographies were examined and how those variations effected the formation and distribution in the hydrothermal system.

The Valles Grande (Valles) caldera in New Mexico is part of the Jemez Mountain volcanic field. This caldera also displays resurgent doming. The system is comprised of a liquid-dominated reservoir with temperatures up to 300 °C and a shallower low-pressure vapour region [Goff, 2002]. This arrangement was expected and achieved by this study's base model conditions (Section 3.3). Further information about the dimensions and

characteristics of these calderas and others reviewed for this study can be found in Appendix 1.

3.3 MODEL DESIGN

3.3.1 Base Model Conditions

The idealized caldera system that forms the basis of this study is illustrated in Figure 3.1 with the corresponding finite-element mesh shown in Figure 3.2.

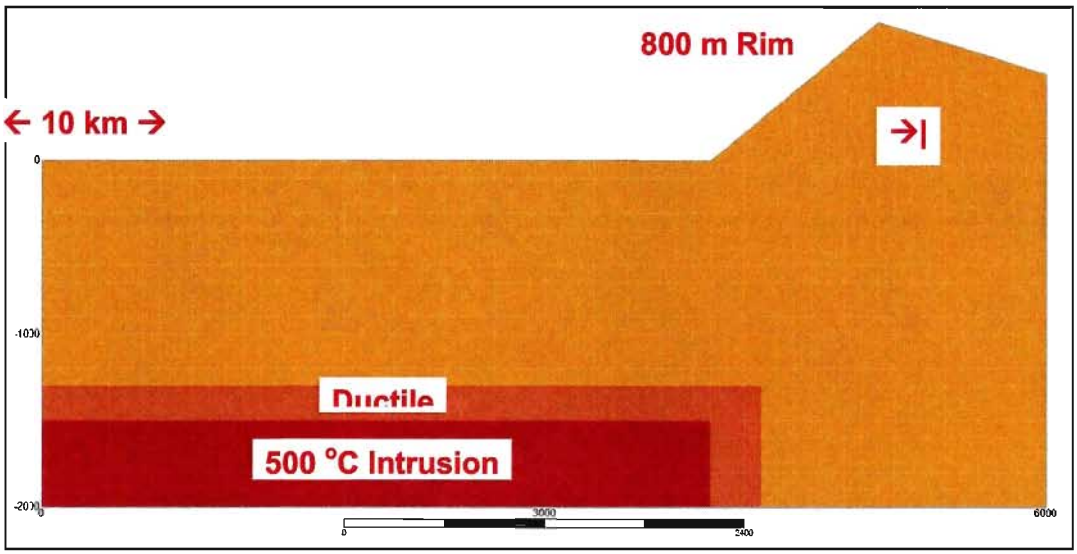


Figure 3.1: Base model conditions consisted of (1) a 10 km-wide caldera with a flat floor and rim height of 800 m; (2) a 500 °C impermeable intrusion 1.5 km below the caldera centre; (3) a regional conductive heat flux twice the continental average (0.10 W/m^2); (4) host rock thermal conductivity of 2.5 W/m °C, density 2650 kg/m^3, and pore fraction 0.05; (5) 250 m ductile region surrounding intrusion.

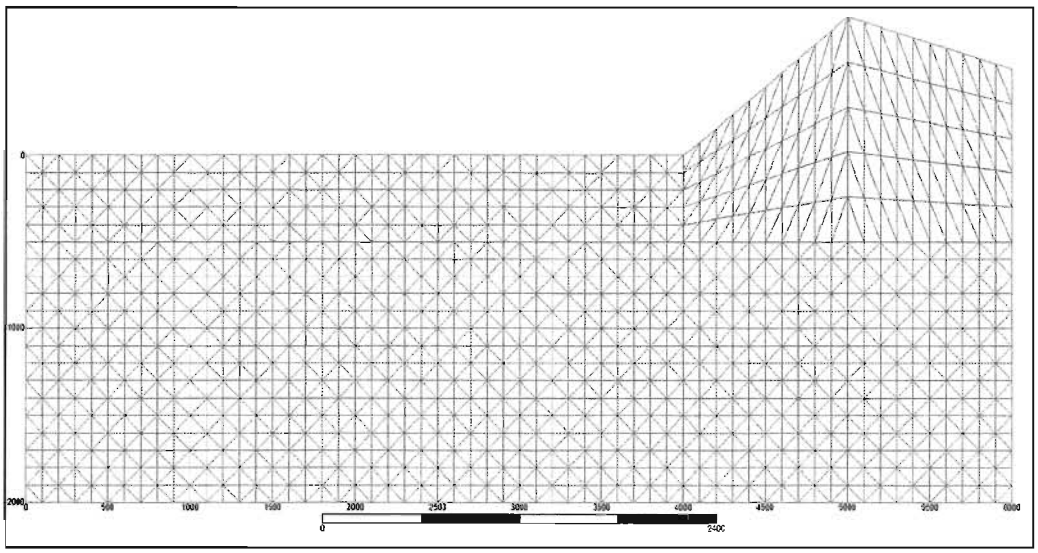


Figure 3.2: Finite-element mesh for base model.

The mesh was composed of 100 m by 100 m linear triangular elements. Three types of linear triangles were used with left-hand triangles for left-facing slopes, right-hand triangles for right-facing, and alternating triangles of the rest of the system. This arrangement was chosen to reduce systematic oscillations in the representations of the fluid flow field in regions where flow direction is governed by topographically-driven flow in the direction of the slope [Cook, S., personal communication, 2008].

In plan view, calderas are generally circular volcanic features as detailed in Section 3.2 and Appendix 1. Thus, a cylindrical coordinate system was used. The centre of the coordinate system corresponds to the centre of the caldera, which corresponds to the left-hand side of Figure 3.1.

Thermal Conditions

The base model (Figures 3.1 and 3.2) consisted of a 10 km-wide caldera with a flat floor and a rim height of 800 m. A 500 °C intrusion was positioned 1.5 km below the caldera centre. The system and intrusion dimensions were chosen because they represent the typical size and depth of the calderas discussed in Section 3.2 and Appendix 1. This intrusion temperature was selected to represent a system in which the intrusion has cooled some since it was first emplaced. This means that the magma would have solidified, and the temperatures are high enough for hydrothermal convection to take place. Section 3.4 illustrates the effects of varying the intrusion temperatures on the hydrothermal convection system and the formation and distribution of steam.

Boundary conditions for the thermal problem include a conductive heat flux for the bottom, insulating sides, and a lapse rate for the surface temperatures. A regional conductive heat flux was set along the lower boundary to 0.10 W/m^2, twice the

continental average, as to be expected in hydrothermal settings with shallow, high temperature intrusions [Della Vedova et al., 2007; Hurwitz et al., 2003]. Temperatures at the surface ranged from 25 °C to 20 °C depending on elevation since temperatures generally decrease with altitude at a rate of approximately 6.5 °C/km [Ahrens, 2007].

Characteristics of the host rock include a thermal conductivity of 2.5 W/m °C, a density of 2650 kg/m^3, and a pore fraction of 0.10. These values were chosen because of the highly fractured, granitic host rock present in these settings (See Appendix 1). A thermal conductivity range of 1.7 W/m °C to 4 W/m °C is given for granitic and porous materials [Engineering Toolbox, 2005]. A value of 2.5 W/m °C was chosen for consistency with previous studies and follows from the amount of fracturing in the system [Andrews & Saar, 2006; Cathles, 1977; Hurwitz et al., 2003].

The density of granitic materials ranges from approximately 2500 kg/m^3 to 2800 kg/m^3 [Engineering Toolbox, 2005; 2700 kg/m^3: Cathles, 1977; 2600 kg/m^3: Giberti, Moreno, & Sartoris, 1984; 2500 kg/m^3: Hurwitz et al., 2003]. An average value of 2650 kg/m^3 was used as a representative value for simulations. Alternative model configurations using densities at the high and low ends of the range showed no measurable effects, indicating that the model results are relatively insensitive to this parameter. Due to the explosive nature of caldera formation, materials in these settings are highly fractured, so a pore fraction of 0.10 was chosen [Andrews & Saar, 2006]. Simulations and their results are outlined in Sections 3.5 and Appendix 2 for the ranges of possible thermal conductivity, density, and pore fraction values.

Fluid Conditions

The intrusion was effectively impermeable and was surrounded by a 250 m-wide ductile region based on and due to elevated temperatures. Because there were not any fractures for fluids to flow through in the ductile region, it was a given a permeability 10^{-3} m^2 less than the system, which was given a homogeneous permeability since the permeability structure cannot be known for this generic model. The type and extent of materials present in caldera-hosted hydrothermal system settings vary significantly between specific sites.

A minimum permeability of 10^{-15} m^2 was required to achieve the target thermal conditions of approximately 220 °C temperatures approximately 300 m below ground surface as observed in active systems such as those discussed previously [Bryan, 2001]. Advective heat transport is effective for permeability values $\geq 10^{-16}$ m^2 [Manning & Ingebritsen, 1999]. The base model resulted in a continuous plume of steam, which originated at the intrusive contact and reached within 300 m of the surface nearly 2 km from the caldera centre (Figure 3.3).

Boundary conditions for fluid flow under steady-state conditions include a fluid flux rate of zero for the bottom and sides of the domain. The top surface has a hydraulic head equal to the topographic elevation (water table), giving the head range for the base model conditions to be from 0 m to 800 m.

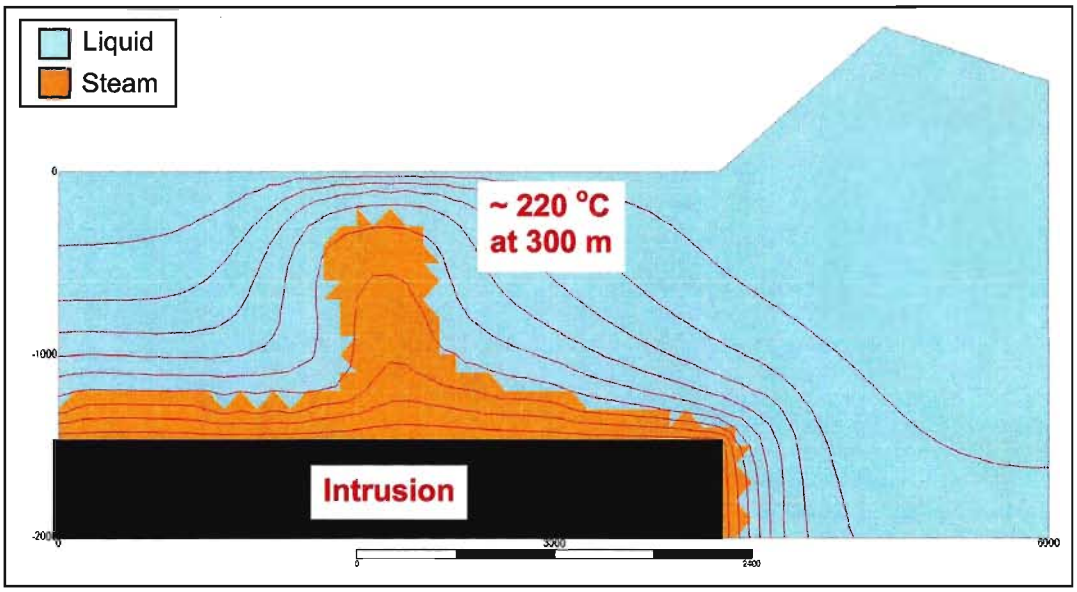


Figure 3.3: Base model solution for a system with homogeneous host rock permeability of 10^{-15} m^2. Target thermal conditions consisted of approximately 220 °C temperatures approximately 300 m below ground surface. Liquid and steam phases and fluid flux vectors are illustrated. Red contours represent 50 °C intervals.

Flow patterns for the base model without and with the intrusion or heat source are illustrated in Figure 3.4. While gravity-driven or topographically-driven flow occurs in both cases due to the topography of the caldera rim, the flow pattern in the system with the heat source is dominated by the effect of buoyancy. Buoyant forces arise due to the density differences between liquid and steam phases causing less dense steam to rise through the crust away from the heat source and more dense liquid to descend.

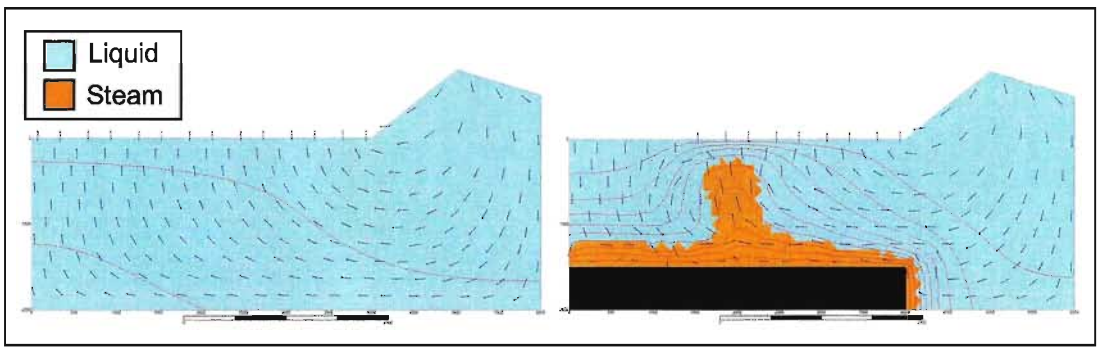


Figure 3.4: Base model solutions for a system without an intrusion or a heat source (left) and with the intrusion (right). Liquid and steam phases and fluid flux vectors are illustrated. Red contours represent 50 °C intervals.

Validation: Mass Balance

To validate that mass is conserved as fluid flows into and out of the steam field, mass flux values were summed around the liquid-steam boundary for the base case scenario run without the latent heat algorithm using the follow raw data obtained from the model:

	Elements		Flux Values				Calculated	
	Steam	Liquid	X (m^2/s)	Z (m^2/s)	Total (m^2/s)	Density (kg/m^3)	Normal Flux (m^2/s)	Mass Flux (kg/s)
1	25	24	1.48E-07	3.18E-08	1.51E-07	53.669	-1.2E-05	-6.44E-04
2	64	63	4.32E-09	4.87E-09	6.51E-09	53.276	4.87E-07	2.59E-05
3	104	103	3.45E-10	5.08E-09	5.09E-09	52.798	5.08E-07	2.68E-05
4	144	143	7.49E-09	-7E-10	7.52E-09	52.599	-7E-08	-3.68E-06
5	184	183	2.69E-09	-8E-10	2.8E-09	52.800	-8E-08	-4.22E-06
6	224	223	1.1E-08	-4.5E-10	1.1E-08	53.181	-4.5E-08	-2.39E-06
7	264	263	7.46E-09	-5E-10	7.48E-09	53.279	-5E-08	-2.66E-06
8	304	303	1.73E-08	1.61E-08	2.36E-08	53.376	1.61E-06	8.59E-05
9	344	343	1.91E-08	1.63E-08	2.51E-08	52.862	1.63E-06	8.62E-05
10	385	384	-1.5E-07	6.74E-08	1.66E-07	53.038	-8.5E-06	-4.51E-04
11	425	424	1.43E-07	1.19E-07	1.86E-07	53.490	-2.3E-06	-1.23E-04
12	465	464	-9.7E-08	1.19E-07	1.54E-07	52.995	2.22E-06	1.18E-04
13	505	504	1.59E-07	1.19E-07	1.98E-07	52.047	-4E-06	-2.08E-04
14	545	544	-8.1E-08	1.19E-07	1.44E-07	51.293	3.83E-06	1.96E-04
15	585	584	2.22E-07	5.58E-08	2.29E-07	50.172	-1.7E-05	-8.53E-04
16	624	623	6.76E-08	3.03E-09	6.77E-08	48.882	3.03E-07	1.48E-05
17	647	646	-6.8E-08	4.65E-07	4.7E-07	17.783	5.33E-05	9.48E-04
18	649	609	-7.7E-08	4.03E-07	4.11E-07	21.524	7.72E-06	1.66E-04
19	651	611	-4.7E-08	3.87E-07	3.9E-07	25.394	4.72E-06	1.20E-04
20	651	652	-4.7E-08	3.87E-07	3.9E-07	25.394	-3.9E-05	-9.90E-04
21	653	652	1.81E-09	3.44E-07	3.44E-07	29.300	3.42E-05	1.00E-03
22	653	654	1.81E-09	3.44E-07	3.44E-07	29.300	-3.4E-05	-9.96E-04
23	664	663	9.27E-08	3.96E-09	9.28E-08	46.813	3.96E-07	1.85E-05
24	686	685	-3.4E-08	4.92E-07	4.94E-07	16.494	4.92E-05	8.12E-04
25	694	654	-1E-07	3.78E-07	3.92E-07	30.953	1.03E-05	3.19E-04
26	697	657	-2.4E-08	3.44E-07	3.45E-07	37.445	2.39E-06	8.95E-05
27	697	698	-2.4E-08	3.44E-07	3.45E-07	37.445	-3.4E-05	-1.27E-03
28	699	698	2.97E-08	2.98E-07	3E-07	41.252	2.69E-05	1.11E-03
29	700	701	2.95E-08	2.69E-07	2.71E-07	43.920	-3E-05	-1.32E-03
30	702	662	8.72E-08	2.6E-07	2.74E-07	48.210	-8.7E-06	-4.19E-04
31	702	701	8.72E-08	2.6E-07	2.74E-07	48.210	2.6E-05	1.25E-03
32	725	685	-5.1E-08	5.22E-07	5.24E-07	14.257	5.09E-06	7.26E-05
33	725	724	-5.1E-08	5.22E-07	5.24E-07	14.257	4.71E-05	6.72E-04
34	765	764	-2.8E-08	5.53E-07	5.53E-07	14.447	5.81E-05	8.39E-04
35	805	804	-6.7E-09	5.54E-07	5.54E-07	14.595	5.47E-05	7.98E-04
36	846	845	1.92E-09	5.15E-07	5.15E-07	17.566	5.15E-05	9.05E-04
37	846	886	1.92E-09	5.15E-07	5.15E-07	17.566	1.92E-07	3.37E-06
38	889	888	7.85E-09	4.67E-07	4.67E-07	23.646	4.75E-05	1.12E-03

39	889	890	7.85E-09	4.67E-07	4.67E-07	23.646	-4.7E-05	-1.11E-03
40	891	890	-5.7E-09	4.3E-07	4.3E-07	28.077	4.24E-05	1.19E-03
41	892	893	-5.6E-09	4.1E-07	4.1E-07	30.991	-4.2E-05	-1.30E-03
42	895	894	-8.5E-08	3.35E-07	3.45E-07	36.492	2.5E-05	9.12E-04
43	897	937	-1.3E-07	3.48E-07	3.71E-07	41.636	-1.3E-05	-5.41E-04
44	898	938	-1.3E-07	3.19E-07	3.43E-07	43.291	-1.3E-05	-5.63E-04
45	940	939	-7.3E-08	2.67E-07	2.77E-07	49.301	2.67E-05	1.32E-03
46	940	980	-7.3E-08	2.67E-07	2.77E-07	49.301	-7.3E-06	-3.60E-04
47	982	981	-9.8E-08	3.53E-08	1.04E-07	48.821	3.53E-06	1.72E-04
48	1022	1021	-8E-08	3.47E-08	8.76E-08	50.807	3.47E-06	1.76E-04
49	1063	1062	-2.2E-07	9.99E-08	2.39E-07	53.753	-1.2E-05	-6.45E-04
50	1104	1103	-6.9E-08	-2.7E-09	6.86E-08	51.376	-2.7E-07	-1.39E-05
51	1144	1143	-7.2E-08	-3.3E-09	7.24E-08	52.501	-3.3E-07	-1.73E-05
52	1184	1183	-7E-08	2.64E-08	7.48E-08	56.750	2.64E-06	1.50E-04
53	1224	1223	-7.9E-08	2.49E-08	8.26E-08	60.762	2.49E-06	1.51E-04
54	1265	1264	-2.5E-07	7.49E-08	2.63E-07	63.628	-1.8E-05	-1.15E-03
55	1306	1305	-6.1E-11	-8.3E-12	6.16E-11	56.484	-8.3E-10	-4.69E-08
56	1346	1345	-7E-11	-9.2E-12	7.04E-11	58.443	-9.2E-10	-5.38E-08
57	1386	1385	-6.2E-11	2.96E-11	6.88E-11	62.956	2.96E-09	1.86E-07
58	1426	1425	-7.3E-11	2.84E-11	7.88E-11	66.449	2.84E-09	1.89E-07
59	1466	1465	-6.8E-11	1.64E-10	1.78E-10	76.687	1.64E-08	1.26E-06
60	1466	1506	-6.8E-11	1.64E-10	1.78E-10	76.687	-6.8E-09	-5.21E-07
61	1507	1506	-8.5E-11	1.71E-10	1.91E-10	66.318	2.57E-08	1.70E-06
62	1547	1546	-2.3E-10	1.71E-10	2.88E-10	71.574	-6.1E-09	-4.37E-07
63	1588	1587	-1.3E-10	3.29E-10	3.56E-10	58.990	3.29E-08	1.94E-06
64	1628	1627	-1.6E-10	3.42E-10	3.76E-10	67.352	3.42E-08	2.30E-06
65	1629	1669	-2.3E-10	4.15E-10	4.76E-10	68.940	-2.3E-08	-1.59E-06
66	1670	1669	-8.4E-11	4.11E-10	4.19E-10	104.32	4.11E-08	4.29E-06
67	1670	1671	-8.4E-11	4.11E-10	4.19E-10	104.32	-4.9E-08	-5.11E-06
68	1673	1672	-4.3E-11	3.48E-10	3.51E-10	94.175	3.05E-08	2.87E-06
69	1674	1675	-4.3E-11	3.11E-10	3.14E-10	96.257	-3.5E-08	-3.37E-06
70	1677	1676	-8.8E-11	2.19E-10	2.36E-10	102.59	1.31E-08	1.34E-06
71	1678	1679	-8.8E-11	1.48E-10	1.72E-10	105.02	-2.4E-08	-2.52E-06
							Sum:	1.89E-03

Fluid flux values at element centres or centroids and were multiplied by the length of the side of the element at the liquid-steam boundary to yield the Normal Flux values. Because the flux values are calculated at element centroids, they represent only approximate flux values for the element boundaries. This influence of the averaging over the area of the element is one of the reasons that the difference between the mass flux out and the mass flux in is not exactly zero (1.89×10^{-03} kg/s). Another reason for the discrepancy is that flux values are used from the steam elements only. Flow in adjacent steam and liquid elements might be different due to variations in fluid properties (i.e.

densities), and that difference will be greater at higher pressures or greater depths. To improve the accuracy of the calculations, the calculations could be made based on values from the "down stream" element (i.e. the element that the fluid is coming from to cross the liquid-steam boundary).

3.3.2 Model Variations

All of the following conditions were simulated without implementing the latent heat algorithm due to issues of convergence and model stability. The results for the simulations that showed a significant impact on the formation and distribution of steam in the system are outlined in Sections 3.4. The remaining results and simulations are presented for reference in Appendix 2. Outlined here are the ranges of values that were tested for each of the parameters in the system.

Basal Heat Flux

Heat flux values tested ranged from 0.05 W/m^2, the average continental value [Della Vedova et al., 2007; Hurwitz et al., 2003] to 0.15 W/m^2, which is higher than the continental average as a result of the high temperature intrusion.

Intrusion Characteristics – Depth

In addition to a base intrusion depth of 1.5 km, simulations were run for intrusions with tops at depths of 2 km, 2.5 km, 3 km, and 5 km below ground surface.

Intrusion Characteristics – Shape

To test the effects of the shape of the intrusion being modelled, five variations, illustrated in Figure 3.5, were made to the rectangular box-shaped intrusion used in the base model: (1) Increasing depth of intrusion by 100 m every 100 m between 3.5 km and 4 km from caldera centre, (2) Increasing depth by 100 m every 200 m between 2 km and

4 km from caldera centre, (3) Increasing depth by 200 m every 200 m between 2 km and 4 km from caldera centre, (4) Increasing depth by 100 m every 1 km between the caldera centre and 4 km from the caldera centre, and (5) Increasing depth by 200 m every 1 km between the caldera centre and 4 km from the caldera centre.

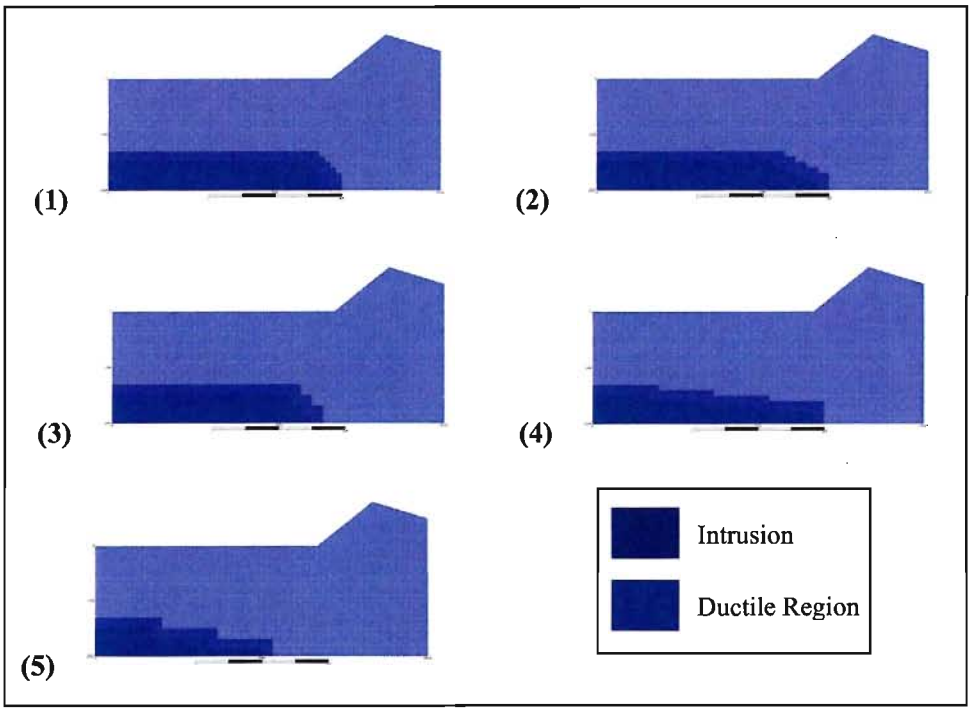


Figure 3.5: Variations in intrusion shape: (1) Increasing depth of intrusion by 100 m every 100 m between 3.5 km and 4 km from caldera centre, (2) Increasing depth by 100 m every 200 m between 2 km and 4 km from caldera centre, (3) Increasing depth by 200 m every 200 m between 2 km and 4 km from caldera centre, (4) Increasing depth by 100 m every 1 km between the caldera centre and 4 km from the caldera centre, and (5) Increasing depth by 200 m every 1 km between the caldera centre and 4 km from the caldera centre.

Intrusion Characteristics – Temperature

The range of intrusion temperature values modelled was from 300 °C to 1000 °C, including temperatures of 400 °C, 450 °C, 550 °C, 600 °C, and 700 °C. The lower temperatures represent systems in which the intrusion has cooled to varying degrees since emplacement. The higher temperatures represent less cooling or more recent emplacement of the intrusion.

Intrusion Characteristics – Width

The width of the intrusion was varied between 5 km (2.5 km from caldera centre to intrusion edge) and 9 km (4.5 km from caldera centre to intrusion edge).

Mesh (Element) Size

To test the model for sensitivity to element size, an alterative mesh with 50 m by 50 m elements was also used for a simulation of base model conditions. Smaller elements or finer meshes were not used because a greater computational effort would have been required, while larger elements or coarser meshes were not used because there would have been a reduction in the accuracy of the results.

Permeability

Advective heat transport is effective for permeability values $\geq 10^{-16}$ m^2 [Manning & Ingebritsen, 1999]. Values of host rock permeability were varied between 10^{-18} m^2 and 10^{-14} m^2 while the permeability for the ductile region (10^{-18} m^2) and the intrusion (10^{-22} m^2) were held constant. Values for the ductile region were varied between 10^{-21} m^2 and (10^{-15} m^2) while the permeability for the host rock (10^{-15} m^2) and the intrusion (10^{-22} m^2) were held constant.

Pore Fraction

Systems with pore fraction values of 0.05, 0.15, 0.20, and 0.30 were simulated. These are reasonable values given the fractured, granitic materials of such a caldera-hosted hydrothermal system [Freeze & Cherry, 1979].

System Size

Caldera systems can be larger than the 10 km-wide base model that was simulated, so the base conditions were expanded to 15 km-wide and 20 km-wide systems also.

Thermal Conductivity

A thermal conductivity range of 1.7 W/m °C to 4 W/m °C is given for granitic and porous materials [Engineering Toolbox, 2005]. The range simulated in this study was from 2.0 W/m °C to 4.0 W/m °C, with 0.5 W/m °C intervals including the base conditions, and additional simulations for 2.25 W/m °C and 2.75 W/m °C thermal conductivity values.

Topography – Resurgent Dome

The base model conditions include a flat caldera floor. Resurgent domes with maximum heights of 100 m, 200 m, and 300 m were simulated for domes extending to 2.5 km, 3 km, 3.5 km, and 4 km on either side of the caldera centre (i.e. 5 km, 6 km, 7 km, and 8 km diameter domes).

Topography – Rim Height

Variations on the height of the caldera rim were simulated. First a system with no rim, no topography, was simulated. Then rim heights of 100 m, 300 m, 500 m, 700 m, 900 m, and 1000 m were modelled.

3.4 MODEL RESULTS

3.4.1 Effect of Latent Heat

The primary results of this study involve the comparison of the base model conditions when simulated with and without the latent heat algorithm, which was developed for Part 2 of this document. The results of simulating the base model conditions without the latent heat algorithm were presented in Section 3.3.

At this regional scale, modelling base model conditions with the latent heat algorithm included in the calculations results in instabilities. At the liquid-steam boundary there is a large amount of averaging in the liquid elements. There are differences in the temperature gradients in the liquid and steam fields as illustrated in Figure 3.6. This leads to oscillations as the model iterates to try to find a solution.

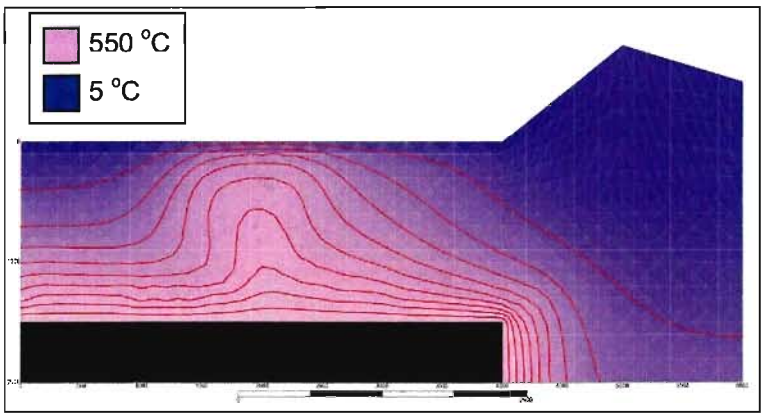


Figure 3.6: Plot of element temperatures for base model conditions. Differences in temperature gradients in liquid and steam fields lead to oscillations in the model. Red contours represent 50 °C intervals.

Head and temperature residuals identical to those used for models without the latent heat algorithm were used when the models were first run with the latent heat algorithm. Only an approximate solution could be reached. As the oscillations occur, numerical artefacts (e.g., negative temperatures near the surface, a "bulls-eye" of

temperature contours) were observed in the solution as the number of iterations is increased. Thus, for comparison purposes, models were run with 5 iterations, which yielded a thermal solution residual for the mesh with element size 100 m by 100 m of 282.28 °C and 255.43 °C for the 50 m by 50 m element mesh (Figure 3.7).

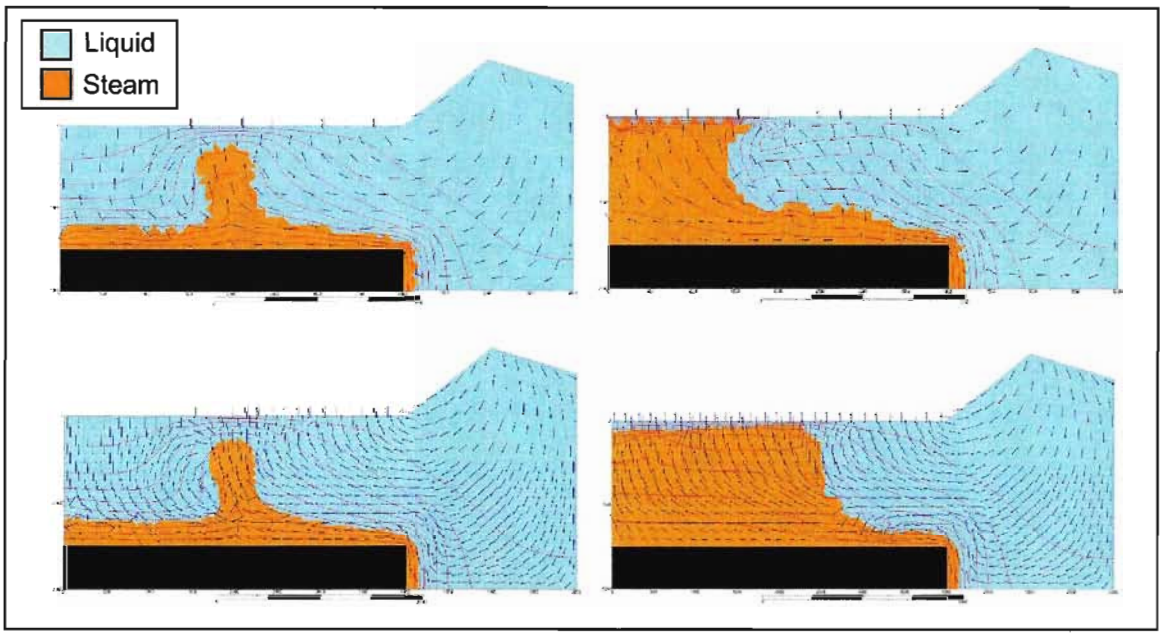


Figure 3.7: Results of simulations of base model conditions without latent heat (left) and with latent heat (right) for 100 m by 100 m elements (top) and 50 m by 50 m elements (bottom).

Increasing the initial temperature residuals did not eliminate the oscillations. To ensure that error compounding was not occurring for non-convergent solutions due to the oscillations and the large number of iterations, the iteration limits were reduced to 10, 15, 20, and 25 iterations. Figure 3.8 illustrates the approximate solutions obtained with those iteration limits. There were no significant differences in the general pattern of steam distribution between the results for the different iteration limits.

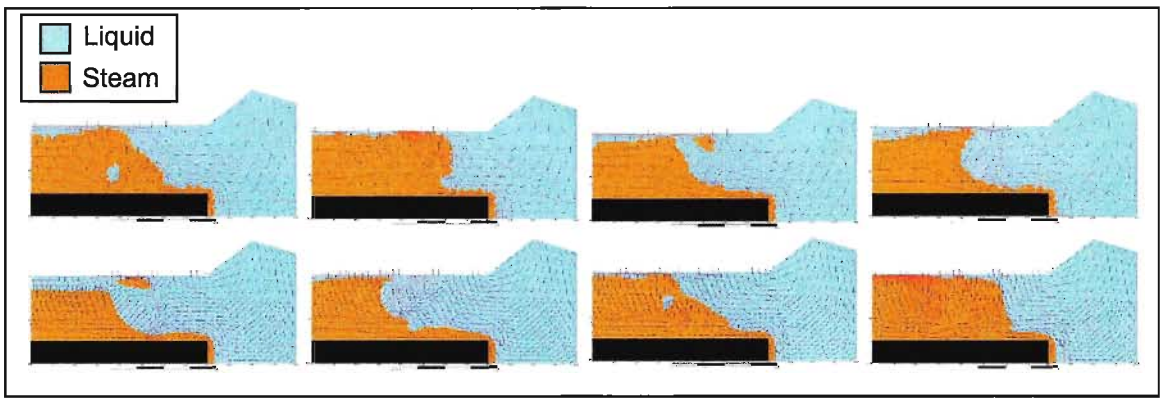


Figure 3.8: Approximate solutions for 100 m by 100 m elements (top) and 50 m by 50 m elements (bottom) with iteration limits of 10, 15, 20, and 25 iterations.

To address non-convergence for the models run with the latent heat algorithm, the approximate solutions that have been calculated can be used as a starting point for mesh refinement. Mesh refinement would reduce property averaging in the liquid elements. Since each element can have only one set of temperature and pressure values, and, therefore, can only represent one phase at a time, larger elements require averaging properties over a greater area than do smaller elements. To minimize the additional computational effort required by refining the mesh, the mesh only needs to be refined in the regions where the liquid-steam boundary occurs in the approximate solution.

Based on the results presented in Part 2, it was expected that the liquid-steam boundary would be displaced in the direction of flow with greater displacement occurring at shallower depths (lower pressures), in regions of smaller thermal gradients, and in more permeable materials (higher flux rates). For the base model conditions, the host rock was considered homogeneous with a permeability of 10^{-15} m^2, which in the column models of Part 2 produced displacement between 17 m and 1322 m depending on temperature and pressure conditions. Rim topography and the thermal gradient also influenced flux rates.

3.4.2 Implications for Steam Distribution in Natural Systems

Although only an approximate solution could be determined for models run with the latent heat algorithm presented in Part 2, the results are considered acceptable for the purposes of this study as discussed in Section 3.4.1. Based on comparisons made between models run with and without the latent heat algorithm, it is evident that accounting for the thermal effects of latent heat is important for determining the volume and distribution of steam in natural systems. Depending on temperature and pressure conditions, steam might be tens to hundreds of metres away from where a model that does not account for latent heat indicates. Thus, it is also vital that when modelling a specific hydrothermal site, the temperature and pressures conditions as well as the flow field (i.e. direction and magnitude of fluid flux) are well known, as discussed in Part 2.

3.4.3 Effects of Parameter Variations on Steam Formation and Distribution

In addition to the comparison of simulations run without and with the latent heat algorithm presented in Section 3.4.1, several general observations can be made based on the effects of varying certain model parameters. Those observations of variations that showed a significant impact on the formation and distribution of steam in the system are presented here. The remaining results and simulations are presented for reference in Appendix 2.

This sensitivity analysis was carried out using models run without simulating the latent heat effects due to convergence problems that can introduce thermal artefacts into the models. Since these artefacts could potentially mask the effects of the other parameters that were varied, the latent heat effects were omitted. While these results, therefore, do not represent the actual distribution of steam, they do represent the general

patterns and magnitude of the effects of varying particular parameters. The actual distribution of steam would result from the combination of the effects of varying the particular parameter and the effects of latent heat on the liquid-steam boundary.

Intrusion Characteristics – Depth, Shape, and Width & System Size

In general, the larger, the deeper, and the hotter the intrusion the longer the geothermal system can exist [Cathles, 1977; Cathles, Erendi, & Barrie, 1997]. All three cases result in situations in which more heat is available to be dissipated from the intrusion to drive the hydrothermal convection. Results of model simulations outlined below are consistent with these findings.

As intrusion depth, and therefore confining pressures, increase, steam plumes shift towards the caldera centre. Deeper intrusions (2 km below ground surface or deeper) produced supercritical fluids along the intrusion contact. The deeper the intrusion, the more steam and supercritical fluids were produced and convected upwards away from the intrusion. Figure 3.9 illustrates the distribution of steam for a system with base model conditions and an intrusion 3 km below ground surface.

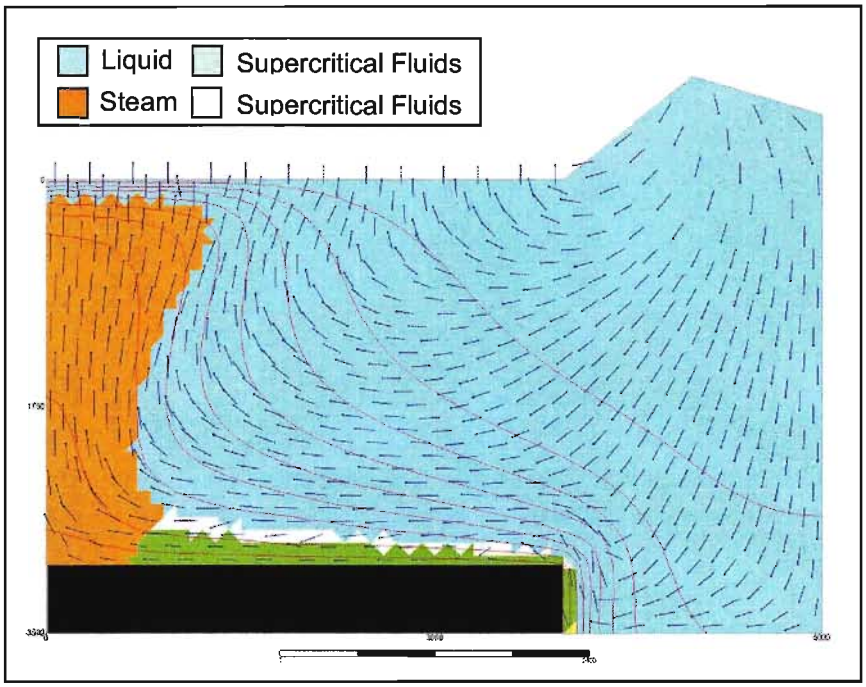


Figure 3.9: Steam distribution for base model conditions with a deeper (3 km below ground surface) intrusion. With greater depths and, therefore, greater pressure, supercritical fluids exist along the intrusion contact in addition to steam.

As intrusion width decreased, steam distribution shifted towards the centre of the caldera with larger volumes of steam reaching target depths (300 m below ground surface). Figure 3.10 illustrates this shift as the intrusion is narrowed from 7 km to 5 km wide. Conversely, as illustrated in Figure 3.11, as the intrusion width increased, steam plumes moved away from the centre of the caldera and an additional plume also originating at the intrusion contact and reaching target depths was formed over the caldera centre. The formation of the additional plume at the caldera centre or the development of a convection cell between plumes is a result of temperature-controlled variations in fluid density [Energy and Geoscience Institute (EGI), 2001].

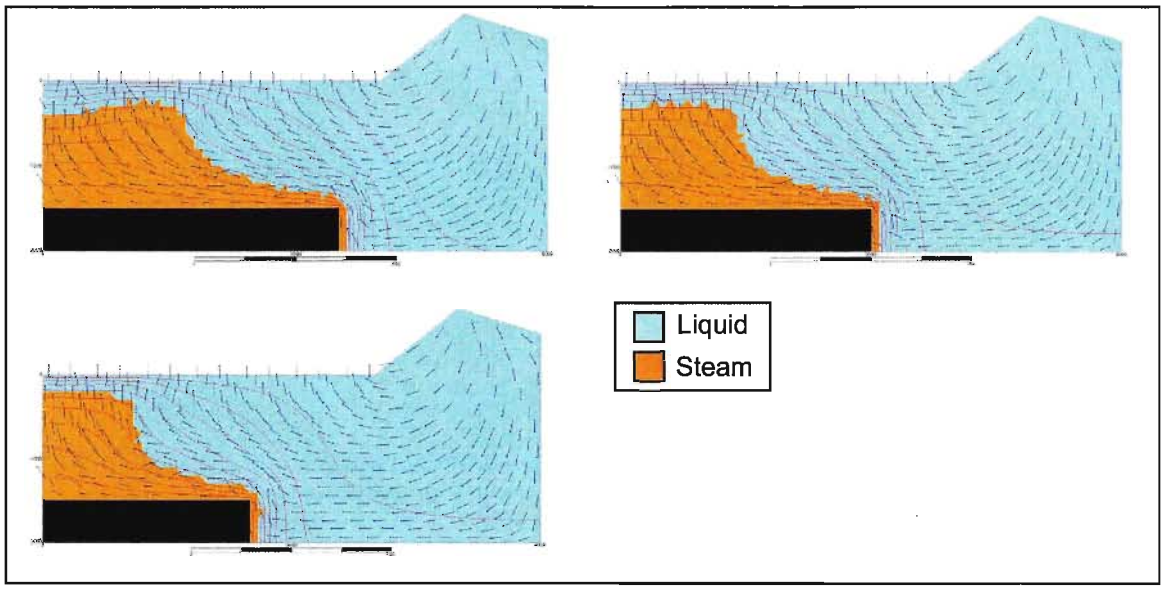


Figure 3.10: As the intrusion becomes narrower, steam distribution is shifted towards the caldera centre creating one large steam plume originating from the intrusion contact and reaching depths of approximately 300 m.

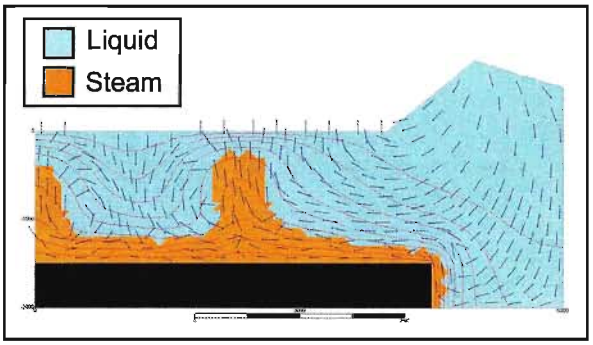


Figure 3.11: As the intrusion becomes wider, the steam plumes still originating from the intrusion contact and reaching depths of approximately 300 m, shift away from the caldera centre. An additional steam plume forms at the centre of the caldera also originating from the intrusion contact and reaching nearly the same depth of 300 m.

Altering the shape of the intrusion from the "box" used in the base model produced results that combine the effects of changing the depth and width of the intrusion. This is expected because the stair-step effect of changing the edge of the intrusion increased the depth and reduced the width of portions of the intrusion to varying

degrees as illustrated in Figure 3.5. The results for two of the variations of intrusion shape are illustrated in Figure 3.12.

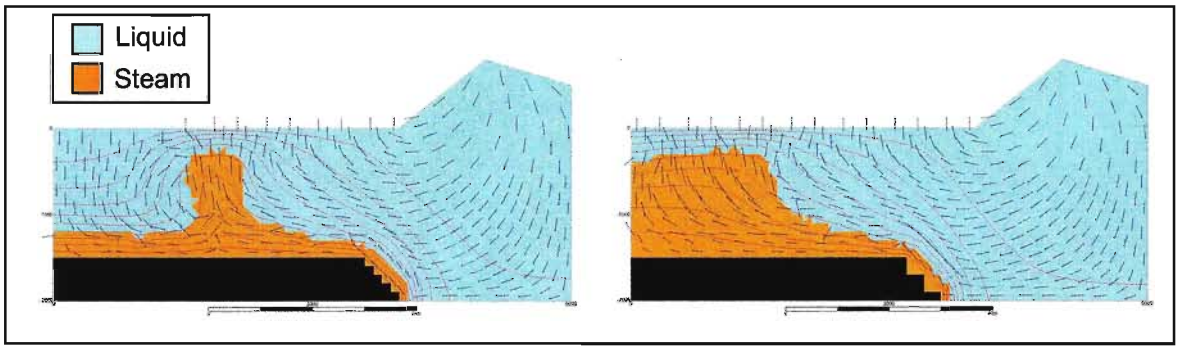


Figure 3.12: Alterations to the shape of the intrusion produce results comparable with changing its depth and width.

Increasing the size of the entire system meant an increase in the size and depth of the intrusion. Both of these increases in intrusion characteristics lead to greater amounts of heat to be dissipated from the intrusion. Figure 3.13 illustrates a system twice the size of the base model. This larger system produces larger volumes of steam than the base model as well as supercritical fluids, which are not present in the base model.

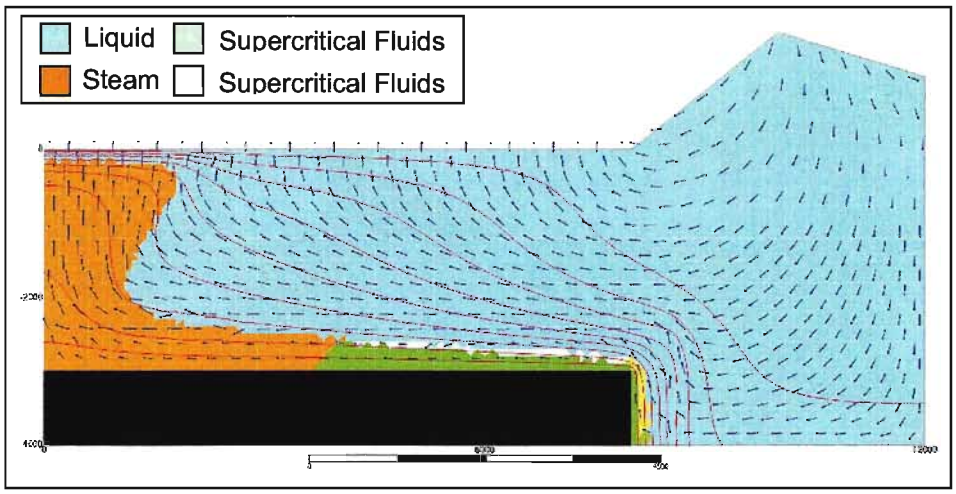


Figure 3.13: Fluid flow patterns and steam distributions for a system twice the size of the base model.

Intrusion Characteristics – Temperature

Intrusion temperatures dictate the amount of heat available to be dissipated from the intrusion to drive the hydrothermal convection. Figures 3.14 and 3.15 illustrate this trend for extremes of intrusion temperatures, 300 °C and 1000 °C. A 300 °C intrusion (Figure 3.14) does not yield hydrothermal convection steam in the system. On the other extreme, a 1000 °C intrusion (Figure 3.15) produces a steam plume originating at the intrusion contact, reaching target depths of 300 m below ground, and spanning nearly the entire width of the intrusion.

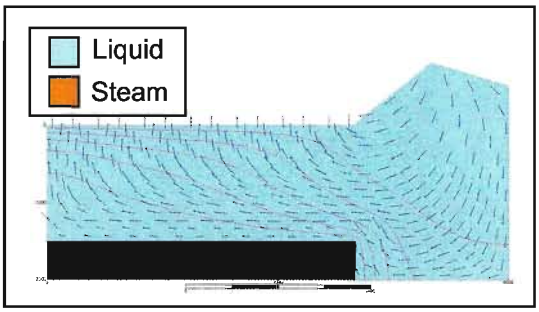


Figure 3.14: Fluid flow pattern for base model conditions with a 300 °C intrusion.

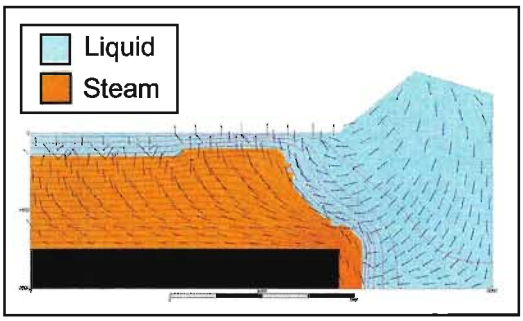


Figure 3.15: Fluid flow pattern and steam distribution for base model conditions with a 1000 °C intrusion.

Figure 3.16 illustrates the evolution of the system as intrusive temperatures are varied from 450 °C to base conditions (500 °C) to 550 °C. With an intrusion at 450 °C a single plume of steam from the intrusion contact to approximately 300 m below ground

surface is produced approximately half way from the caldera centre to the side end of the intrusion. An increase in intrusion temperature just 50 °C to base model conditions produces a wider steam plume further away from the caldera centre. A further increase in intrusion temperature again yields wider plume of steam, and also produces a steam plume at the centre of the caldera from the intrusion contact to target depths.

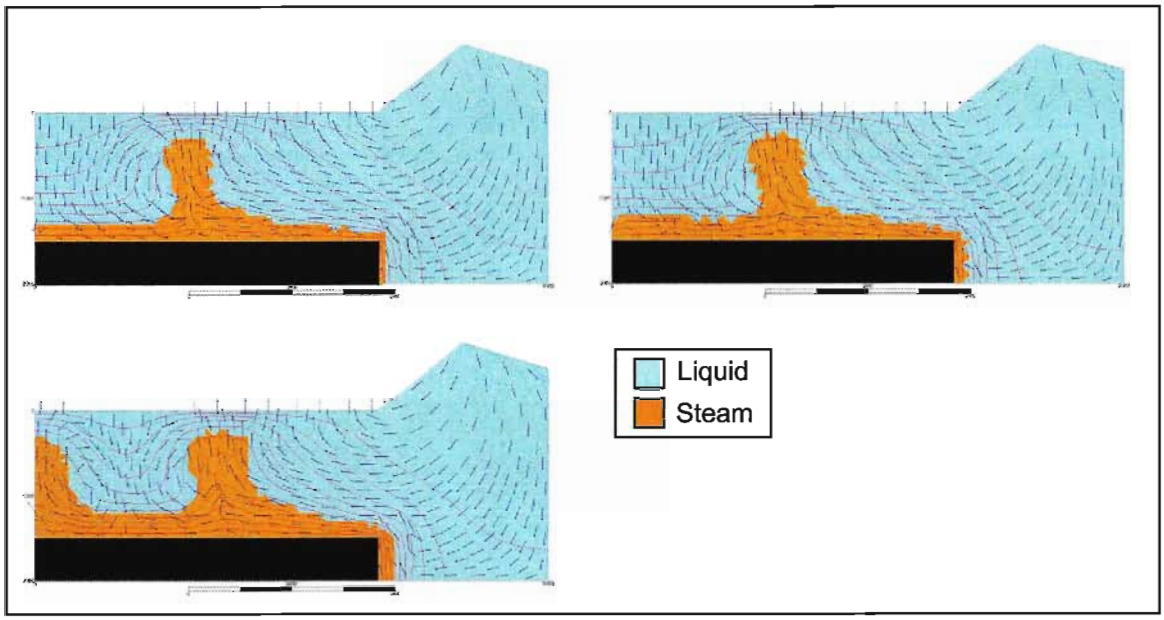


Figure 3.16: Variations in fluid flow patterns and steam distribution as a result of changing temperature of the intrusion. From left to right, a 450 °C, base model conditions (500 °C intrusion), and a 550 °C intrusion.

Mesh (Element) Size

Using a mesh with smaller elements did not effect the formation or distribution of steam in the system. As seen in Figure 3.17, the boundary between liquid and steam is smoother for the mesh with smaller elements than for the base mesh used for this study.

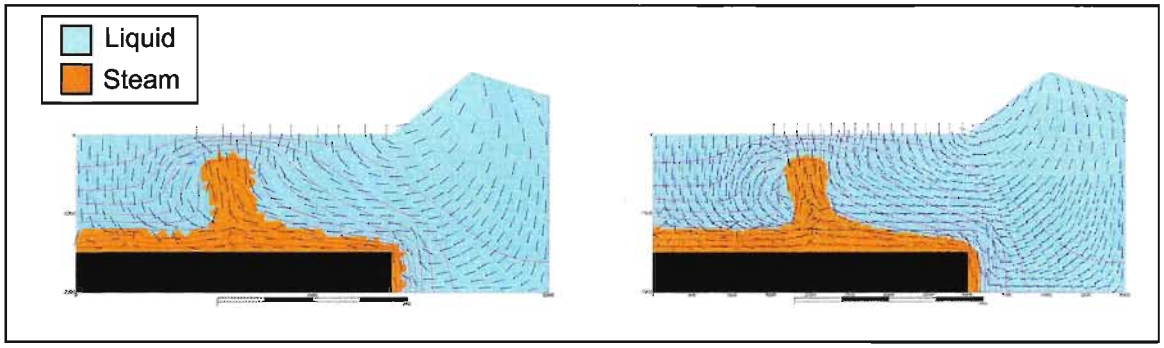


Figure 3.17: Base model conditions were simulated using meshes with element sizes 100 m by 100 m (left, standard for this study) and 50 m by 50 m (right).

Permeability

Host rock permeability values less than that of the base model (10^{-15} m^2), produced a band of steam over the length of the intrusion, but not reaching target depths (Figure 3.18) because there is not enough advection. Host rock permeability values greater than that of the base model allowed for greater amounts of advection, shifting the volume of steam towards the caldera centre and upwards towards the surface (Figure 3.19). This mushrooming plume reached shallower depths than those reached in the base case models (< 300 m below ground surface).

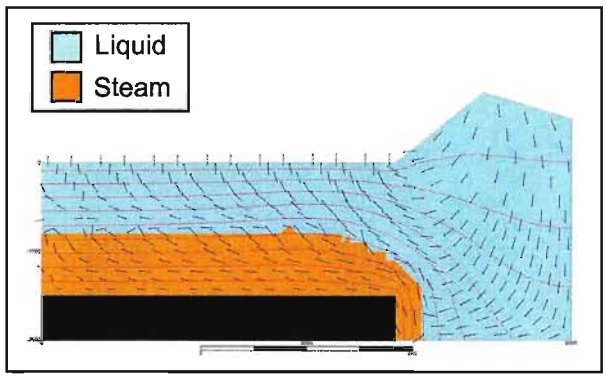


Figure 3.18: Fluid flow patterns and steam distribution for base model conditions with a host rock permeability of 10^{-16} m^2.

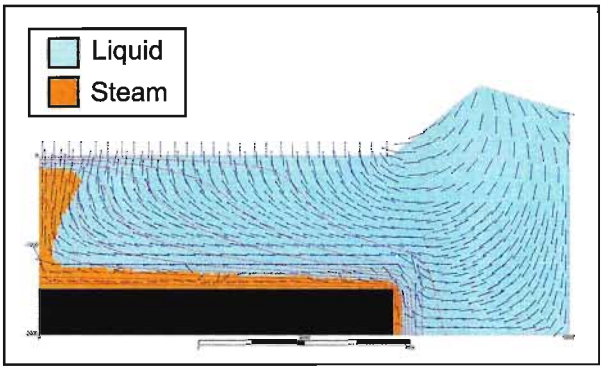


Figure 3.19: Fluid flow patterns and steam distribution for base model conditions with a host rock permeability of 10^{-14} m^2.

Pore Fraction

Pore fractions greater than 0.10 led to the formation of an additional steam plume above the caldera centre. This plume also originated at the intrusion contact, but was not as shallow as target conditions, reaching up to depths of 400 m below ground surface with a pore fraction of 0.15. Results of a simulation with pore fraction of 0.20 is illustrated in Figure 3.20.

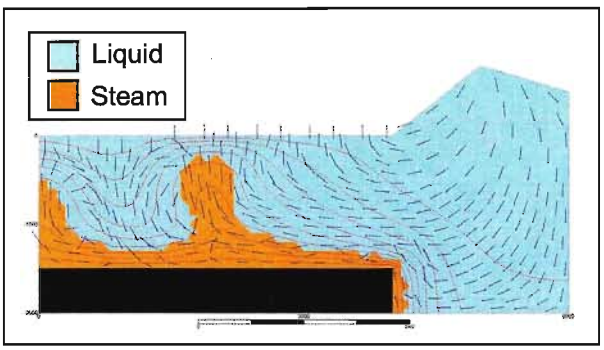


Figure 3.20: Fluid flow patterns and steam distribution for base model conditions with a pore fraction of 0.20.

Topography – Resurgent Dome

Resurgent domes with a maximum height of 300 m were simulated. Increasing dome height from a flat floor redistributed steam towards the caldera centre, narrowing

the peripheral plumes. Figure 3.21 illustrates this trend for domes extending to 4 km on either side of the caldera centre (i.e. 8 km diameter domes).

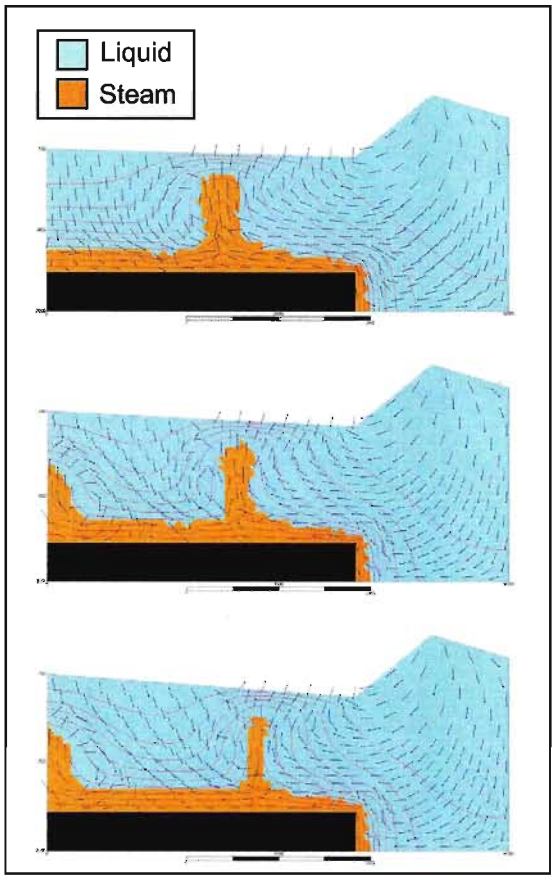


Figure 3.21: Variations in fluid flow patterns and steam distribution with a progression of resurgent dome heights. Systems, from left to right, with 100 m dome height, 200 m dome height, and 300 m dome height.

Topography – Rim Height

With no rim or surface topography, steam is produced at the caldera centre and at approximately 3.5 km from the centre in plumes originating from the intrusion contact and reaching target depths. Adding topography for the caldera rim influences the pattern of hydrothermal convection, gradually shifting the steam from the plume over the caldera centre towards the peripheral steam plumes, which gradually shift inwards towards the caldera centre. The distribution of steam shifts because the fluid flow is able to overcome

the buoyancy effect in the centre of caldera. Figure 3.22 illustrates these changes in changes in steam distribution from no rim to rim heights up to 1000 m.

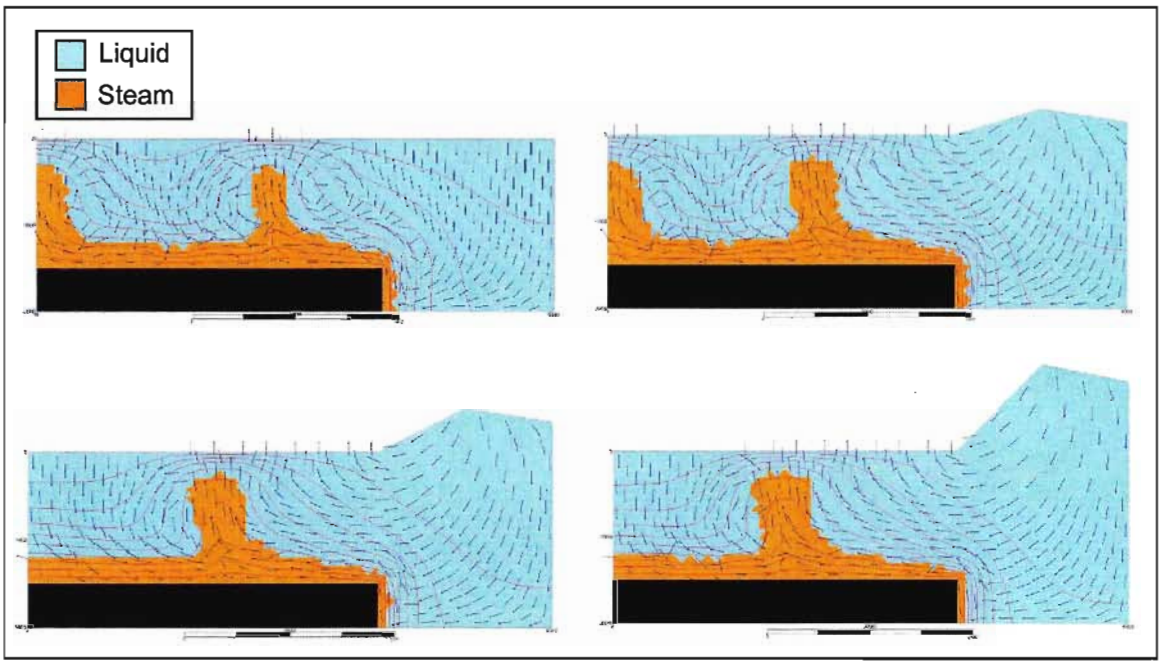


Figure 3.22 Variations in fluid flow patterns and steam distribution with a progression of caldera rim heights. Systems, from left to right, with no rim, with a 300 m rim height, a 500 m rim height, and a 1000 m rim height.

3.5 SUMMARY AND CONCLUSIONS

A series of steady-state numerical simulations were run on a hypothetical caldera-hosted system based on the characteristics of a representative suite of calderas. Base model conditions consisted of a 10 km-wide caldera with a flat floor and topographic rim height of 800 m; a 500 °C intrusion located 1.5 km beneath the caldera centre; a background conductive heat flux two times the continental average (0.10 W/m^2); host rock thermal conductivity of 2.5 W/m C, with density 2650 kg/m^3 and pore fraction (0.10). The intrusion was modelled as impermeable with a 500 m wide surrounding region given a permeability magnitude 10^{-3} m^2 less than the system to represent a ductile region produced by the elevated temperature of the intrusion. The remainder of the system was modelled with homogenous permeability.

All models were run using cylindrical coordinates to better represent the geometry of caldera systems. For these conditions, a permeability of 10^{-15} m^2 was required to achieve conditions temperatures of around 220 °C at approximately 300 m below ground surface as observed in active systems. This model also resulted in a continuous steam plume originating at the intrusive contact that reached within 300 m of the surface along the edges of the caldera (approximately 2 km from caldera centre).

Results were compared for simulations run before and after the addition of the latent heat algorithm as presented in Part 2. Although only an approximate solution could be determined for models run with the latent heat algorithm, the results are valid for the purposes of this study since there were no significant differences in the general pattern of steam distribution between the results for the different iteration limits (i.e., error compounding was not occurring for non-convergent solutions due to the oscillations and

the large number of iterations). Mesh refinement along the liquid-steam boundary from the approximate solutions obtained would address the issue of non-convergence.

As expected, the liquid-steam boundary was displaced in the direction of flow. Greater displacement could be observed at shallower depths (lower pressures). Displacement was also affected by varying thermal gradients and caldera rim topography due to their influence on the flux rate at the given homogenous permeability.

Parameter variations yielded information about the sensitivity of the system to a number of properties and conditions. This information is valuable for the design of future site-specific studies. This study, which used 100 m by 100 m elements, has shown that the size of the intrusion or heat source should be known within 500 m, and its temperature within 50 °C. The permeability should be known within an order of magnitude, however, the structure of both the permeability and porosity can be known in more detail for site-specific studies. By modelling fractures and heterogeneities in a caldera-hosted hydrothermal system, for example, can indicate where surface features might occur. The topography of the system, namely the resurgent doming and rim height should be known within 100 m.

The ability to model phase change conditions accurately and to understand the factors that control steam formation and distribution is essential for determining effective methods of locating and preserving or exploiting resources associated with modern and ancient hydrothermal systems. With the latent heat algorithm included, numerical models can be used to simulate both the spatial and temporal evolution of site-specific hydrothermal systems at any scale. This will allow it to be used as a predictive tool for determining where and when steam will form within the system, which is important for

locating ore deposits associated with ancient systems and geothermal reservoirs in active systems. Without the latent heat calculations, results could lead exploration geologists to miss ore deposits by tens or even hundreds metres depending on pressure and temperature conditions. In addition to providing insight into accessing such important resources, modelling exercises can inform preservation efforts for systems such as the Yellowstone caldera system where national park borders do not necessarily match up with the system boundaries.

3.6 REFERENCES

Ahrens, C. D. (2007). *Meteorology Today: An Introduction to Weather, Climate, and the Environment.* Thomson Brooks/Cole, Belmont, California.

Andrews, J. L. & Saar, M. O. (2006). Coupling and decoupling of heat and helium transport in a geothermal reservoir. *Proceedings from TOUGH Symposium,* Lawrence Berkeley National Laboratory, Berkeley, California, May 15-17, 2006.

Barnes, H. L. (1997). *Geochemistry of Hydrothermal Ore Deposits.* John Wiley & Sons, New York.

Bryan, T. S. (2001). *The Geysers of Yellowstone.* 3rd Edition. University Press of Colorado, Boulder, Colorado.

Cathles, L. M. (1977). An analysis of the cooling of intrusives by ground-water convection which includes boiling. *Economic Geology, 72,* p 804-826.

Cathles, L. M., Erendi, A. H. J., & Barrie, T. (1997). How long can a hydrothermal system be sustained by a single intrusive event? *Economic Geology, 92,* p 766-771.

Chang, R. (2005). *Chemistry*, Eighth Edition. McGraw-Hill Companies, New York.

Christiansen, R. L. (2001). *Geology of Yellowstone National Park: The Quaternary and Pliocene Yellowstone Plateau Volcanic Field of Wyoming, Idaho, and Montana.* Professional Paper 729-G. U.S. Geological Survey, Reston, Virginia.

Della Vedova, B., Vecellio, C., Bellani, S., & Tinivella, U. (2007). Thermal modelling of Larderello geothermal field (Tuscany, Italy). *International Journal of Earth Sciences.*

Edwards, R. & Atkinson, K. (1986). *Ore Deposit Geology.* Chapman and Hal, London.

Energy and Geoscience Institute (EGI). (2001). *Geothermal Energy: Clean Sustainable Energy for the Benefit of Humanity and the Environment.* University of Utah.

Engineering Toolbox. (2005). *The Engineering Toolbox.* [Online Resource]. Retrieved December 4th, 2007 from http://www.engineeringtoolbox.com/.

Freeze, R. A. & Cherry, J. A. (1979). *Groundwater.* Prentice Hall, Inc., Englewood Cliffs, New Jersey.

Giberti, G., Moreno, S., & Sartoris, G. (1984). Evaluation of approximations in modelling the cooling of magmatic bodies. *Journal of Volcanology and Geothermal Research, 20,* p 297-310.

Goff, F. (2002). *Geothermal Potential of Valles Caldera, New Mexico.* Geo-Heat Center Bulletin.

Hurwitz, S., Kipp, K. L., Ingebritsen, S. E., & Reid, M. E. (2003). Groundwater flow, heat transport, and water table position within volcanic edifices: Implications for volcanic processes in the Cascade Range. *Journal of Geophysical Research, 108,* B12, 2557.

InfoMine Inc. (2008). *Bingham Canyon.* [Online Resource]. Retrieved October 20th, 2008 from http://www.infomine.com/minesite/.

International Geothermal Association (IGA). (2009*). Italy: Electricity Generation.* [Online Resource]. Retrieved July 30th, 2009 from http://iga.igg.cnr.it/index.php.

Lipman, P. W. (1997). Subsidence of ash-flow calderas: relation to caldera size and magma chamber geometry. *Bulletin of Volcanology, 59,* p 198-218.

Lipman, P. W. (2006). Geologic Map of the Central San Juan Caldera Cluster, Southwestern Colorado. *Pamphlet to accompany Geologic Investigations Series I-2799.*

Manning, C. E. & Ingebritsen S. E. (1999). Permeability of the continental crust: Implications of geothermal data and metamorphic systems. *Reviews of Geophysics, 37*, 1, p 127-150.

Nave, C. R. (2005). HyperPhysics. Department of Physics and Astronomy, Georgia State University. [Online Resource]. Retrieved July 30th, 2009 from http://hyperphysics.phy-astr.gsu.edu/hbase/hph.html#hph.

Rio Tinto. (2008). Kennecott Utah Copper. [Online Resource]. Retrieved October 20th, 2008 from http://www.kennecott.com/.

Robb, L. (2005). *Introduction to Ore-Forming Processes*. Blackwell Publishing, Malden, MA.

Sass, J. & Priest, S. (2002). Geothermal California. *Geothermal Resource Council (GRC) Bulletin.*

4. General Discussion and Conclusions

The ability to model phase change conditions accurately and to understand the factors that control steam formation and distribution in hydrothermal systems is essential for determining effective methods of locating, preserving, or exploiting valuable contemporary resources and for understanding the thermal evolution of their ancient counterparts. To investigate the thermal effects of latent heat and simulate steam formation conditions in hydrothermal settings an algorithm was added to the existing finite-element numerical modelling software, Aquarius. The latent heat algorithm was validated using a series of column models and then applied to a generic caldera-hosted hydrothermal system based on a representative suite of calderas.

Part 2 outlined the procedure involved in development and implementation of a latent heat model to account for the thermal effects of phase changes between liquid and steam. The corresponding algorithm was coded in the C# programming language and added to the Aquarius model. A copy of the code is provided in Appendix 4. Validation of the latent heat algorithm involved the use of one-dimensional column models, which allowed for investigation of the effects that certain parameters and conditions have on the magnitude of the effects of latent heat. The effects of flow rate, thermal gradient, and position along the two-phase curve were investigated and the following trends were observed:

- As the flow rate (permeability) is increased, the magnitude of displacement of the liquid-steam boundary from an initial position determined without accounting for latent heat is increased until the effects of the system boundary were greater than the effects of the latent heat.

- As the thermal gradient is increased, the magnitude of displacement is reduced until the effects of the system boundary were greater than the effects of the latent heat. In flowing from steam to liquid, for example, the larger difference in temperature means that the liquid is cool enough to absorb all of the energy released from the phase change without reaching the boiling point.
- As pressure is increased or the position along the two-phase curve approaches the critical point, the magnitude of displacement is reduced due to decreasing values of latent heat energy until the effects of the system boundary were greater than the effects of the latent heat.

In Part 3 the latent heat algorithm was applied to a regional scale model to further investigate the effects of latent heat. A series of steady-state numerical simulations were run on a hypothetical caldera-hosted system based on the characteristics of a representative suite of calderas (e.g., Yellowstone caldera in Yellowstone National Park, USA, Creede caldera in Colorado, and Valles Grande caldera in New Mexico). Results were compared for simulations run without and with the latent heat algorithm. Additional parameter variations yielded information about the sensitivity of the system to a number of properties and conditions.

Only approximate two-dimensional solutions could be obtained for models run with the latent heat algorithm. However, the results are considered reasonable first approximations for the purpose of this study because there were no significant differences in the general pattern of steam distribution between the results for the different iteration limits (i.e., error compounding was not occurring for non-convergent solutions due to the oscillations and the large number of iterations). Results of one-dimensional column

models constructed with 2 m scale elements suggest that refinement of the mesh along the liquid-steam boundary in the approximate solutions could potentially solve the issue of non-convergence by eliminating the averaging of fluid properties across elements with steep thermal gradients downstream of the liquid-steam interface.

As expected from the trends observed in Part 3, the liquid-steam boundary was displaced in the direction of flow with greater displacement occurring at shallower depths (lower pressures). This is a direct consequence of the latent heat for water, which increases with decreasing pressure along the two-phase curve. Displacement was also affected by varying thermal gradients and caldera rim topography due to their influence on the flux rate at the given homogenous permeability. The results of this comparison indicate that without the latent heat calculations, model solutions could lead exploration geologists to miss ore deposits by tens or even hundreds metres depending on pressure and temperature conditions.

This study, which used 100 m by 100 m elements, has shown that the size of the intrusion or heat source should be known within 500 m, and its temperature within 50 °C. The permeability should be known within an order of magnitude, however, the structure of both the permeability and porosity can be known in more detail for site-specific studies. Modelling fractures and heterogeneities in a caldera-hosted hydrothermal system, for example, can indicate where surface features might occur. The topography of the system, namely the resurgent doming and rim height should be known within 100 m.

With the latent heat effects included, numerical models can be used to simulate both the spatial and temporal evolution of site-specific hydrothermal systems at any scale. This will allow it to be used as a predictive tool for determining where and when

steam will form within the system, which is important for locating ore deposits made in ancient systems and geothermal reservoirs in modern systems. Modelling exercises can inform preservation efforts for systems such as the Yellowstone caldera system where national park borders do not necessarily match up with the system boundaries.

APPENDIX 1: CALDERA GEOLOGY AND HYDROTHERMAL FEATURES

A1.1 Caldera Formation

In this study, hydrothermal activity was modelled in a caldera setting. Calderas form by the collapse of land after a volcanic eruption. The following diagrams (Figure A1.1) illustrate the caldera formation process in a granitic system like the one at Yellowstone in Wyoming, USA. To begin, the magma chamber causes the Earth's crust to bulge at the surface (Figure A1.1, stage 1). The bulging or uplift in turn causes the crust to crack and fracture in a circular or ring-like pattern (Figure A1.1, stage 2). The magma chamber will then begin to empty through the cracks and fractures (Figure A1.1, stage 3). Molten rock in these systems are silica-rich with a high water content. These materials are viscous and, because the trapped gases decompress quickly when they reach the surface, highly explosive.

With the magma chamber emptied or nearly so, it can no longer support the weight above it, and the crust collapses leaving a circular, approximately symmetrical feature in place of the magma chamber (Figure A1.1, stage 4). Subsequent magma intrusions can cause further uplift or bulging of the caldera floor known as a resurgent dome.

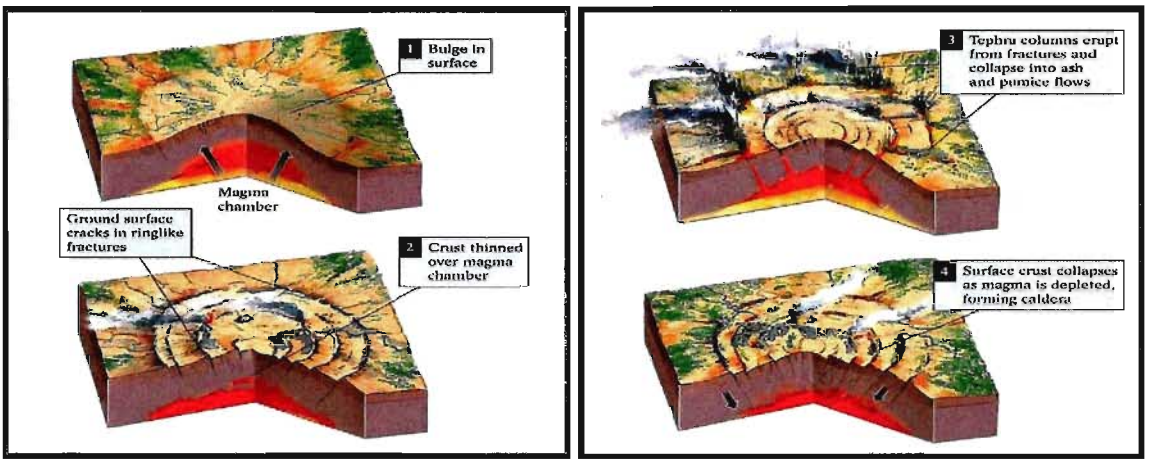


Figure A1.1: Formation of a caldera in four stages [Chernicoff & Whitney, 2002].

A1.2 Hydrothermal Features

Evidence of boiling or steam formation in active hydrothermal systems exists in the form of surface features such as steam vents, mud pots, hot springs, and geysers. Steam vents, or fumaroles, occur when boiling occurs below ground and the steam is able to rise through vents to the surface. Mud pots form when that rising steam heats overlying water. They are essentially fumaroles drowning in mud. When the rising hot water reaches the surface hot springs can be observed. If that rising hot water enters a pressurized reservoir first, a geyser erupts when the pressure is suddenly released.

Evidence of boiling or steam formation in ancient hydrothermal systems exists in the form of ore deposits such as copper and gold [Barnes, 1997; Cathles, 1977; Geological Survey of Japan, 2000]. The majority of ore deposits either are a direct product of the concentration arising from the circulation of hydrothermal fluids or have been significantly modified by them [Robb, 2005]. Four changes in a hydrothermal solution can lead to mineral precipitation under specific conditions: temperature changes, pressure changes, chemical reactions between solution and surrounding rock, and fluid mixing [Barnes, 1997; Robb, 2005]. Any one change or a combination of several changes can lead to precipitation, which occurs because such changes reduce the solubility of the metals in the hydrothermal fluid [Robb, 2005].

These features and the larger hydrothermal systems they are a part of occur in association with geologically recent or current volcanic activity. Figure A1.2 illustrates the overlap between regions of geothermal activity and tectonic plates. Hydrothermal convection can occur at zones of subduction (e.g., volcanic island arcs), regions of plate

stretching or spreading (e.g., mid-ocean ridges), and "hot spots" or mantle plumes (e.g., Yellowstone caldera system).

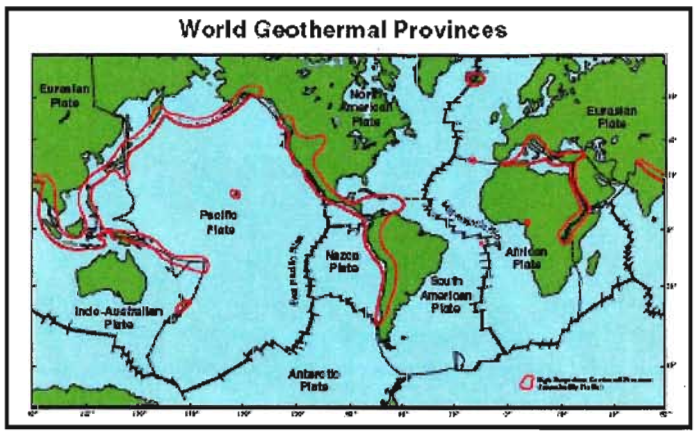


Figure A1.2: Regions of geothermal activity and plate tectonics. High temperature geothermal provinces are outlined in red [EGI, 2001].

A1.3 Caldera Descriptions

<table>
<tr><td colspan="2">Caldera: Crater Lake</td></tr>
<tr><td>Location: Oregon, USA</td><td>Dimensions:
Diameter: 8 km by 10 km
Depth: 1200 m [Cockell, Koeberl, & Gilmour, 2006]</td></tr>
<tr><td colspan="2">Description: The Crater Lake caldera formed due to collapse of Mount Mazama, a composite volcano, along ring fractures. Additional volcanic activity caused the formation of volcanic features on the caldera floor, which is composed of collapse breccias and intracaldera tuff. Lake sediments are 20 – 40 m thick. The lake itself, the deepest in the United States [U.S. DOI, 2008], is 622 m deep and thermally stratified. [Cockell, Koeberl, & Gilmour, 2006]</td></tr>
<tr><td colspan="2">Geologic History: Approximately 6850 years ago [Cockell, Koeberl, & Gilmour, 2006; USGS, 2008] 55 km^3 of hot gas and dense rock erupted during the caldera-forming event. During the single-vent phase, pumice and ash were deposited over more than 1 million km^2. When the eruption column collapsed it generated four pyroclastic flow units. With the magma chamber partially emptied, the caldera collapse occurred along the circular or arcuate ring fractures, depositing pumiceous ignimbrite and coarse lithic breccia in and around the caldera.

Additional explosive phreatic eruptions resulted in Wizard Island (800-900 years ago, [USGS, 2008]), which is a cinder cone that projects 233 m above lake's surface and Merriam Cone, which is entirely submerged [Cockell, Koeberl, & Gilmour, 2006; U.S. Department of the Interior, 2008]. In the past 5000 years there has been no evidence of continued volcanic activity at Mount Mazama, but future activity could be expected, likely beneath the surface of the lake [U.S. Department of the Interior, 2008].</td></tr>
</table>

Caldera: Long Valley	
Location: California, USA	**Dimensions:** Diameter: 16 km by 32 km The caldera floor ranges in elevation from 2000 m in the Eastern half to 2600 m in the Western half. The caldera walls rise steeply to elevations of 3000 – 3500 m on all sides except in the East and Southeast where the floor rises 150 m to merge with the Volcanic Tableland at 2300 m in elevation [USGS, 2008].
Description: The Eastern half of the caldera is dominated by Lake Crowley and Long Valley, which is sage and grass covered. The Western half is hillier and heavily forested [USGS, 2008]. Groundwater flow is driven by topography and heat from underlying magma. Recharge to the system is primarily from snow-melt in the highlands around the Western and Southern rims of the caldera. Meteoric water infiltrates the crust to depths of a few kilometres where it is heated by hot rock near the intrusions to temperatures of 220 °C or more. Upflow occurs in the Western end along steeply inclined fractures to depths of 1 – 2 km. Hydrothermal fluids also flow laterally down the hydraulic gradient around the resurgent dome [USGS, 2008].	
Geologic History: The Long Valley caldera resulted from a violent eruption of 600 km^3 of magma in the form of pyroclastic flow and ash 760 000 years ago. These materials cooled to form the Bishop Tuff and caused 2-3 km of subsidence of the magma chamber to form the present day caldera. This caldera is part of the volcanic system, which includes the Mono-Inyo craters and stretches about 50 km form Mammoth Mountain to Mono Lake. Although major volcanic activity has not occurred since the formation of Mammoth Mountain about 50 000 years ago, some mild activity was observed as recently as 600 years ago. Even today the system is considered active because of frequent earthquake activity [USGS, 2008].	

Caldera: Valles Grande	
Location: Jemez Mountain Volcanic Field New Mexico, USA	**Dimensions:** Diameter: 20-24 km [Treiman, 2003], 22 km [Goff, 2002] Depth: 300 m [Treiman, 2003]
Description: Valles Grande caldera can be referred to as a "trap door" caldera as it is deeper in the East than on the West. ***Convective*** heat flow in the caldera is >5000 mW/m^2, while deep ***conductive*** flow outside the caldera is up to 400 mW/m^2. Fifty kilometres around the caldera springs, a well, and an aquifer contain gaseous fluids with variable chemistries. The caldera contains a liquid-dominated reservoir with temperatures up to 300 °C. It is recharged by local meteoric water. Above the reservoir is a low-pressure vapour cap [Goff, 2002].	
Geologic History: The Jemez Mountain Volcanic Field (JMVF) erupted calc-alkaline basalt, andesite, dacite, and rhyolite from 13 Ma to 55 ka, with the Valles caldera forming at about 1.2 Ma. The Valles caldera has high-silica rhyolite ignimbrites and post-caldera rhyolitic products. The JMVF is located where the Jemez Lineament (JL) and the western margin of the Rio Grande Rift (RGR) intersect. [Goff, 2002]	

<table>
<tr><td colspan="2">Caldera: Indonesian Volcanoes: Batur, Ijen, Krakatau, Tambora, Tengger</td></tr>
<tr><td>Location: Indonesia</td><td>Dimensions:
Batur volcano at centre of two concentric calderas
Outer Diameter: 10 km by 13 km
Inner Diameter: 7.5 km
Ijen (Kendeng) caldera
Diameter: 20 km
Krakatau
Diameter: 7 km
Tambora
Diameter: 6 km
Depth: 120 m
Tengger volcanic complex five overlapping stratovolcanoes, each truncated by calderas
Largest Calera Diameter: 16 km
Most Recent Caldera Diameter: 9 km by 10 km (Sandsea caldera)
[USGS, 2008]</td></tr>
<tr><td colspan="2">Description/Geological History: Batur: The inner caldera formed during the emplacement of Bali ignimbrite between 23 670 and 28 500 years ago. The inner caldera lies beneath Lake Batur on the south-east side of the larger, outer caldera [USGS, 2008].

Ijen: In this caldera is a group of small stratovolcanoes and a 1 km wide turquoise-coloured, acid crater lake [USGS, 2008].

Krakatau: In 416 AD collapse of the Krakatau edifice led to a caldera with a diameter of 7 km. Lavas and pyroclastics accumulated and Rakata, Danan, and Perbuwatan volcanoes formed. In 1883, another eruption destroyed Danan and Perbuwatan. At that time the post-collapse cone of Anak Krakatau was formed, becoming a site of frequent eruptions since 1927 [Smithsonian Institute, 2009; USGS, 2008].

Tambora: More than 43 000 years ago an earlier caldera was formed, but it was later covered in by late-Pleistocene lava flows. In 1815, an eruption of more than 150 km^3 of tephra was the source of history's largest explosive eruption causing global cooling and creating the present day caldera. Minor lava domes and flows extruded on the caldera floor [USGS, 2008].

Tengger: Five overlapping stratovolcanoes, dating from 820 000 years ago, each truncated by a caldera made up the volcanic complex of the early Pleistocene. Along the flanks of the caldera are lava domes, pyroclastic cones, and a maar [Smithsonian Institute, 2009; USGS, 2008].</td></tr>
</table>

Caldera: Aniakchak	
Location: Alaska, USA	**Dimensions:** Diameter: 10 km Depth: 1 km (maximum) Rim Elevation: 1341 m to 610 m **[USGS, 2008]**
Description: Vent Mountain is the largest cone rising 430 m above the caldera floor with a 2.5 km diameter. Ash flows were non-welded and highly mobile filling adjacent glacial valleys to a depth of at least 75 m [USGS, 2008].	
Geological History: This caldera was formed approximately 3400 years ago in one of more than 40 explosive eruptions to occur in the last 10000 years. Pyroclastic cones, tuff, maars, and lava domes mark the ice-free caldera floor [Smithsonian Institute, 2009].	

Caldera: Creede Caldera	
Location: San Juan Caldera Cluster (Bachelor, Creede, La Garita, San Luis) Colorado, USA	**Dimensions:** *Creede caldera* Diameter: 24 km [Lipman, 1997]
Description/Geologic History: The San Juan volcanic field is a prime example not only of large-volume continental volcanism with at least 22 major ash flow sheets (150 – 5000 km^3 each), but also of continental ore deposition of epithermal base- and precious-metal veins [Lipman, 2006]. Eruptions occurred 30 -26 million years ago, with the Creede caldera forming 26.5 million years ago from the eruption of Snowshoe Mountain tuff. Of all the calderas in the San Juan cluster, Creede is best preserved within the largest of the calderas, La Garita (35 by 75 km), which collapsed in three segments [Lipman, 1997, 2006]. Maximum diameter of calderas in the San Juan cluster ranges from 10 km to 75 km [USGS, 2008]. The Creede caldera is a resurgent feature [Lipman, 1997, 2006].	

Caldera: Yellowstone Caldera	
Location: Yellowstone National Park, USA	**Dimensions:** 72 km by 48 km [Smith & Siegel, 2000] 85 km by 45 km [Christiansen, 2001]
Description/Geologic History: Three major caldera eruptions shaped the Yellowstone caldera observed at the present time. The first occurred two million years ago when 2500 km3 of lava and ash erupted into a caldera 80 km by 65 km. Hot ash, rock, and pumice covered the ground up to 97 km around the caldera. Another eruption 1.3 million years ago ejected 280 km^3 of hot ash and rock, and created a 24 km wide caldera. The third eruption 630000 years ago formed the 72 km by 48 km caldera seen today. A volume of 1000 km^3 hot rock ash was erupted. Following the last caldera eruption two resurgent domes, were formed. The first, the Sour Creek dome is approximately 300 m high. The Mallard Lake dome began to bulge 150000 years ago [Smith & Siegel, 2000].	

REFERENCES

Barnes, H. L. (1997). *Geochemistry of Hydrothermal Ore Deposits*. John Wiley & Sons, New York.

Cathles, L. M. (1977). An analysis of the cooling of intrusives by ground-water convection which includes boiling. *Economic Geology, 72*, p 804-826.

Cockell, C. Koeberl, C. & Gilmour, I. (2006). *Biological Processes Associated with Impact Events.* Springer, Heidelberg, Germany.

Energy and Geoscience Institute (EGI). (2001). *Geothermal Energy: Clean Sustainable Energy for the Benefit of Humanity and the Environment.* University of Utah.

Geological Survey of Japan. (2000). *Epithermal gold deposits, geothermal systems and volcanoes: Efficient gold exploration through applied research.* [Online Resource]. Retrieved December 6th, 2007 from http://www.gsj.go.jp/dMR/Jikken/Epithermal.html.

Goff, F. (2002). *Geothermal Potential of Valles Caldera, New Mexico.* Geo-Heat Center Bulletin.

Lipman, P. W. (1997). Subsidence of ash-flow calderas: relation to caldera size and magma chamber geometry. *Bulletin of Volcanology, 59*, p 198-218.

Lipman, P. W. (2006). Geologic Map of the Central San Juan Caldera Cluster, Southwestern Colorado. *Pamphlet to accompany Geologic Investigations Series I-2799.*

Robb, L. (2005). *Introduction to Ore-Forming Processes.* Blackwell Publishing, Malden, MA.

Smith, R. B. & Siegel, L. J. (2000). *Windows into the Earth: The Geologic History of Yellowstone and Grand Teton National Parks*. Oxford University Press, New York.

Smithsonian Institute (National Museum of Natural History). (2009). *Global Volcanism Program.* [Online Resource]. Retrieved January 10th, 2009 from http://www.volcano.si.edu/.

U.S. Department of the Interior (DOI). (2008). *National Park Service.* [Online Resource]. Retrieved November 30th, 2008 from http://www.nps.gov/.

U.S. Geological Survey (USGS). (2008). *Volcano Hazards Program.* [Online Resource]. Retrieved November 30th, 2008 from http://www.usgs.gov/.

APPENDIX 2: ADDITIONAL PARAMETER TESTS

This appendix outlines results of additional parameter tests omitted from the body of the thesis. For each simulation all other conditions and parameters were identical to the base model conditions. The only difference is the parameter indicated. For reference, the base results are included here as well.

Base Model Conditions

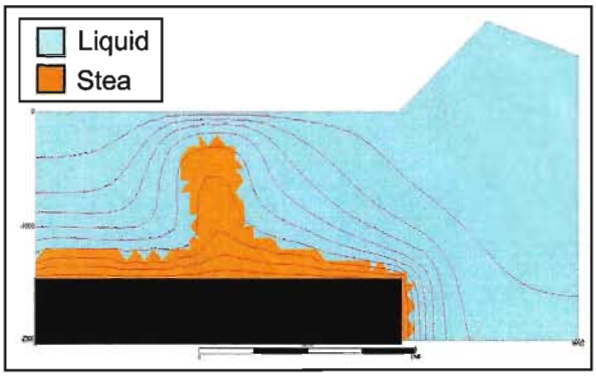


Figure A3.1: Base Model Conditions. A 10 km-wide caldera with rim height of 800 m and flat floor was modelled with an 8 km-wide, 500 °C intrusion, which was 1.5 km below ground surface and surrounded by 250 m ductile region. The following properties were given: a density of 2650 Kg/m^3, a pore fraction of 0.05, a thermal conductivity of 2.5 W/m °C, a basal conductive heat flux of 0.10 W/m^2, and a layered permeability structure (Intrusion: 10^{-22} m^2, Ductile: 10^{-18} m^2, Host Rock: 10^{-15} m^2).

Basal Heat Flux

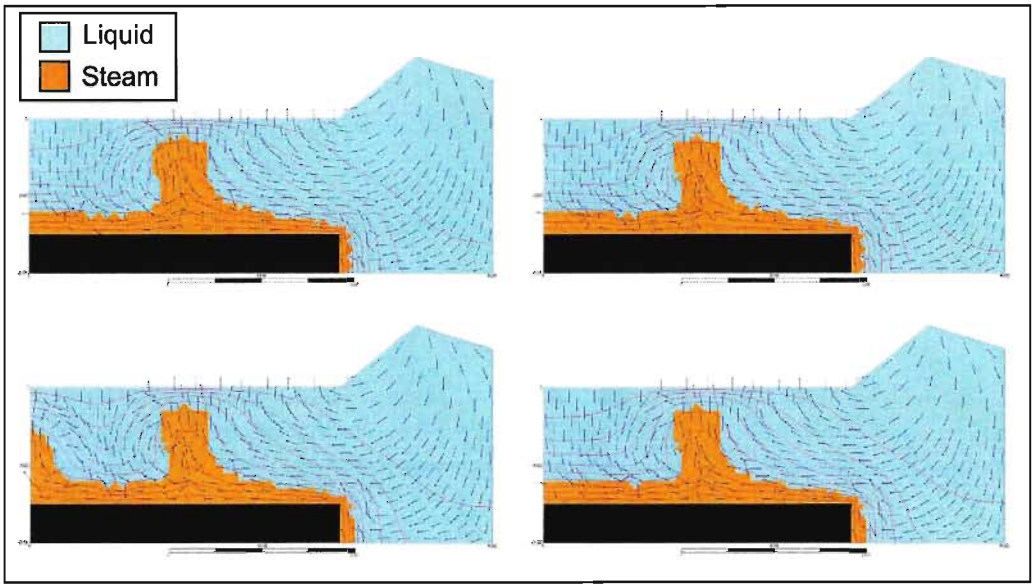


Figure A3.2: Basal Heat Flux: 0.05, 0.075, 0.125, and 0.15 W/m^2.

Intrusion Characteristics – Depth

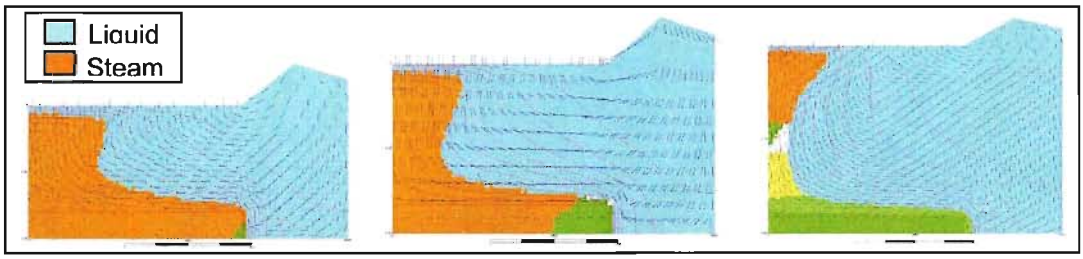


Figure A3.3: Intrusion Characteristics – Depth: 2.0, 2.5, and 5.0 km.

Intrusion Characteristics – Shape

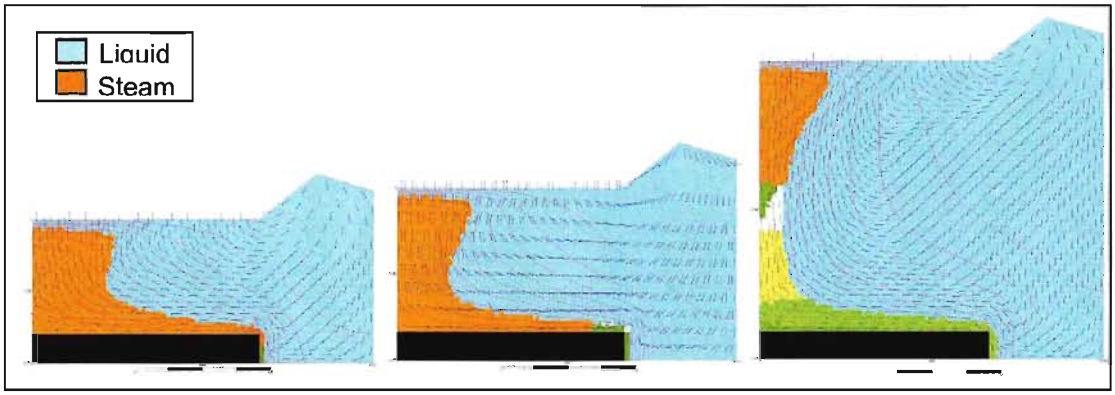


Figure A3.4: Intrusion Characteristics – Shape: Increasing depth by 100 m every 200 m between 2 km and 4 km from caldera centre; increasing depth by 100 m every 1 km between the caldera centre and 4 km from the caldera centre; increasing depth by 200 m every 1 km between the caldera centre and 4 km from the caldera centre.

Intrusion Characteristics – Temperature

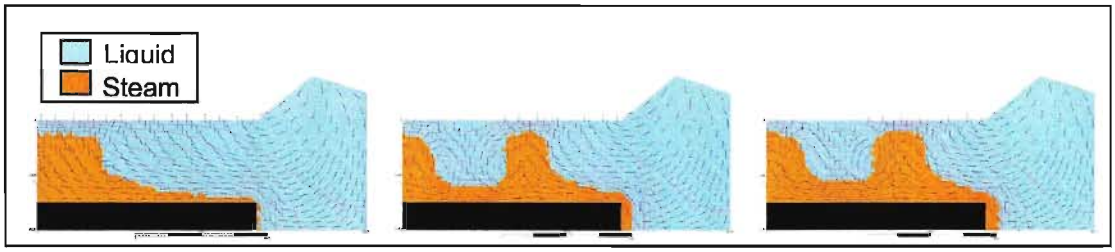


Figure A3.5: Intrusion Characteristics – Temperature: 400, 600, and 700 °C

Permeability

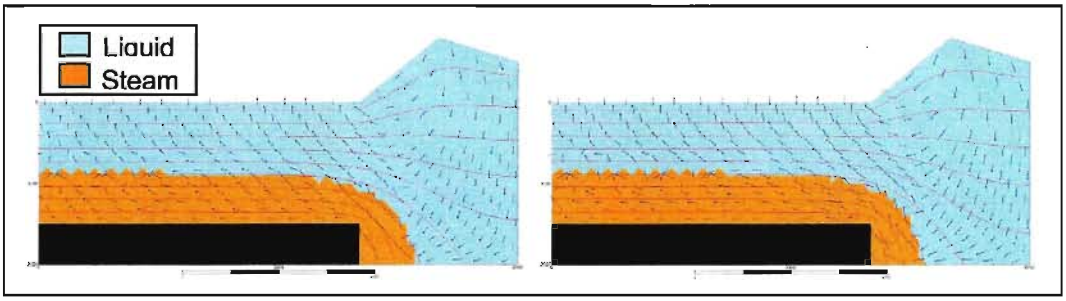


Figure A3.6: Host Rock Permeability: 10^{-18} and 10^{-17} m^2.

Pore Fraction

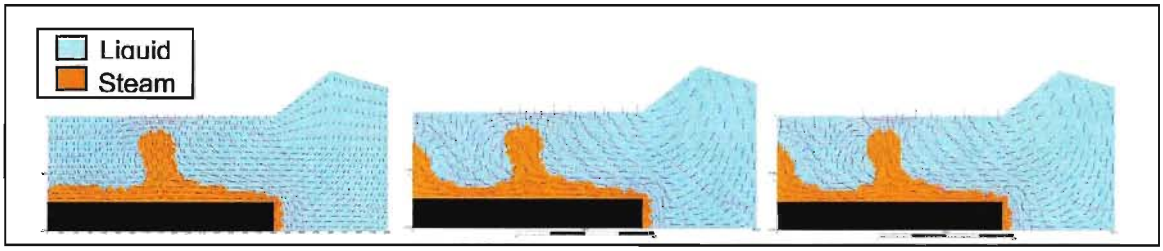


Figure A3.7: Pore Fraction: 0.10, 0.15, and 0.20.

System Size

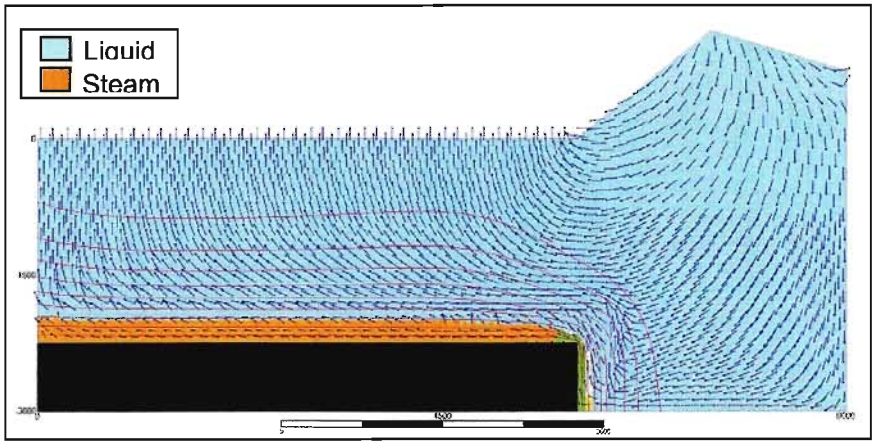


Figure A3.8: System Size: 1.5 times base model. A mesh with element size 50 m by 50 m was used since portions of the mesh were not divisible by 150 m or the base mesh size of 100 m.

Topography – Resurgent Dome

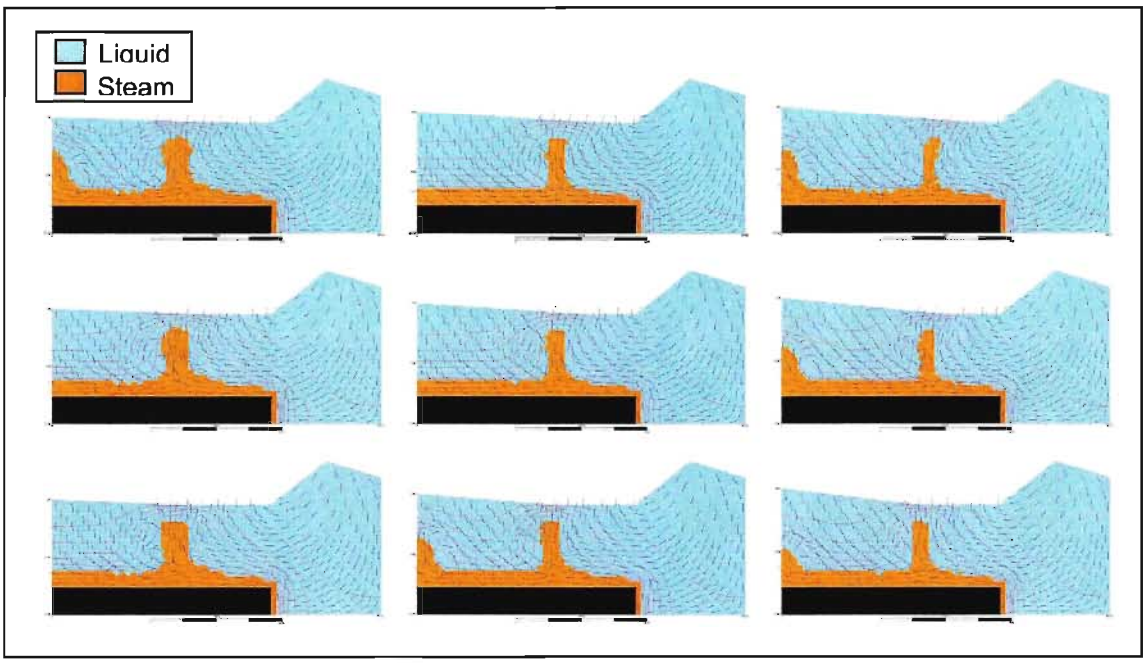


Figure A3.9: Topography – Resurgent Dome: Top Row – 8 km wide, 100, 200 and 300 m high; Middle Row – 7 km wide, 100, 200, and 300 m high; Bottom Row – 6 km wide, 100, 200, and 300 m high.

Topography – Rim Height

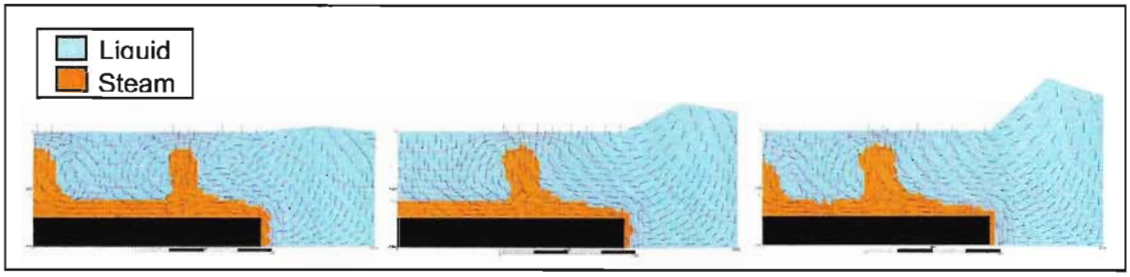


Figure A3.10: Topography – Rim Height: 100, 500, and 900 m.

APPENDIX 3: SATURATION (TEMPERATURE) TABLE WITH ENTHALPY (ΔH)

Included for reference is the Saturation (Temperature) table from *Thermodynamic Properties of Water: Tabulation from the IAPWS Formulation 1995 for the Thermodynamic Properties of Ordinary Water Substance for General and Scientific Use* [Harvey, 1995].

Table 1. Saturation (Temperature)

		Density, kg/m³		Enthalpy, kJ/kg			Entropy, kJ/(kg·K)			Volume, cm³/g	
t, °C	p, MPa	ρ_L	ρ_V	h_L	h_V	Δh	s_L	s_V	Δs	v_L	v_V
0.01	0.000 611 7	999.79	0.004 855	0.00	2500.9	2500.9	0.000 00	9.1555	9.1555	1.000 21	205 991.
1	0.000 657 1	999.85	0.005 196	4.18	2502.7	2498.6	0.015 26	9.1291	9.1138	1.000 15	192 439.
2	0.000 706 0	999.89	0.005 563	8.39	2504.6	2496.2	0.030 61	9.1027	9.0720	1.000 11	179 758.
3	0.000 758 1	999.92	0.005 952	12.60	2506.4	2493.8	0.045 89	9.0765	9.0306	1.000 08	168 008.
4	0.000 813 5	999.93	0.006 365	16.81	2508.2	2491.4	0.061 10	9.0505	8.9894	1.000 07	157 116.
5	0.000 872 6	999.92	0.006 802	21.02	2510.1	2489.0	0.076 25	9.0248	8.9486	1.000 08	147 011.
6	0.000 935 4	999.89	0.007 266	25.22	2511.9	2486.7	0.091 34	8.9993	8.9080	1.000 11	137 633.
7	0.001 002 1	999.86	0.007 757	29.43	2513.7	2484.3	0.106 37	8.9741	8.8677	1.000 14	128 923.
8	0.001 073 0	999.80	0.008 276	33.63	2515.6	2481.9	0.121 33	8.9491	8.8278	1.000 20	120 829.
9	0.001 148 3	999.74	0.008 826	37.82	2517.4	2479.6	0.136 24	8.9243	8.7881	1.000 26	113 304.
10	0.001 228 2	999.65	0.009 407	42.02	2519.2	2477.2	0.151 09	8.8998	8.7487	1.000 35	106 303.
11	0.001 313 0	999.56	0.010 021	46.22	2521.0	2474.8	0.165 87	8.8754	8.7096	1.000 44	99 787.
12	0.001 402 8	999.45	0.010 670	50.41	2522.9	2472.5	0.180 61	8.8513	8.6707	1.000 55	93 719.
13	0.001 498 1	999.33	0.011 355	54.60	2524.7	2470.1	0.195 28	8.8274	8.6321	1.000 67	88 064.
14	0.001 599 0	999.20	0.012 078	58.79	2526.5	2467.7	0.209 90	8.8037	8.5938	1.000 80	82 793.
15	0.001 705 8	999.06	0.012 841	62.98	2528.3	2465.4	0.224 46	8.7803	8.5558	1.000 94	77 875.
16	0.001 818 8	998.90	0.013 645	67.17	2530.2	2463.0	0.238 97	8.7570	8.5180	1.001 10	73 286.
17	0.001 938 4	998.73	0.014 493	71.36	2532.0	2460.6	0.253 43	8.7339	8.4805	1.001 27	69 001.
18	0.002 064 7	998.55	0.015 385	75.54	2533.8	2458.3	0.267 83	8.7111	8.4433	1.001 45	64 998.
19	0.002 198 3	998.36	0.016 325	79.73	2535.6	2455.9	0.282 18	8.6884	8.4063	1.001 64	61 256.
20	0.002 339 3	998.16	0.017 314	83.91	2537.4	2453.5	0.296 48	8.6660	8.3695	1.001 84	57 757.
21	0.002 488 2	997.95	0.018 354	88.10	2539.3	2451.2	0.310 73	8.6437	8.3330	1.002 05	54 483.
22	0.002 645 3	997.73	0.019 448	92.28	2541.1	2448.8	0.324 93	8.6217	8.2967	1.002 28	51 418.
23	0.002 811 1	997.50	0.020 598	96.46	2542.9	2446.4	0.339 08	8.5998	8.2607	1.002 51	48 548.
24	0.002 985 8	997.25	0.021 806	100.65	2544.7	2444.0	0.353 18	8.5781	8.2250	1.002 75	45 858.
25	0.003 169 9	997.00	0.023 075	104.83	2546.5	2441.7	0.367 22	8.5566	8.1894	1.003 01	43 337.
26	0.003 363 9	996.74	0.024 406	109.01	2548.3	2439.3	0.381 23	8.5353	8.1541	1.003 27	40 973.
27	0.003 568 1	996.47	0.025 804	113.19	2550.1	2436.9	0.395 18	8.5142	8.1191	1.003 54	38 754.
28	0.003 783 1	996.19	0.027 269	117.37	2551.9	2434.6	0.409 08	8.4933	8.0842	1.003 82	36 672.
29	0.004 009 2	995.90	0.028 805	121.55	2553.7	2432.2	0.422 94	8.4725	8.0496	1.004 11	34 716.
30	0.004 247 0	995.61	0.030 415	125.73	2555.5	2429.8	0.436 75	8.4520	8.0152	1.004 41	32 878.
31	0.004 496 9	995.30	0.032 102	129.91	2557.3	2427.4	0.450 52	8.4316	7.9810	1.004 72	31 151.
32	0.004 759 6	994.99	0.033 868	134.09	2559.2	2425.1	0.464 24	8.4113	7.9471	1.005 04	29 526.
33	0.005 035 4	994.66	0.035 717	138.27	2561.0	2422.7	0.477 92	8.3913	7.9134	1.005 37	27 998.
34	0.005 325 1	994.33	0.037 651	142.45	2562.8	2420.3	0.491 55	8.3714	7.8799	1.005 70	26 560.
35	0.005 629 0	993.99	0.039 674	146.63	2564.5	2417.9	0.505 13	8.3517	7.8466	1.006 05	25 205.
36	0.005 947 9	993.64	0.041 790	150.81	2566.3	2415.5	0.518 67	8.3321	7.8135	1.006 40	23 929.
37	0.006 282 3	993.29	0.044 001	154.99	2568.1	2413.1	0.532 17	8.3127	7.7806	1.006 76	22 727.
38	0.006 632 8	992.92	0.046 311	159.17	2569.9	2410.8	0.545 62	8.2935	7.7479	1.007 13	21 593.
39	0.007 000 2	992.55	0.048 723	163.35	2571.7	2408.4	0.559 03	8.2745	7.7154	1.007 50	20 524.
40	0.007 384 9	992.18	0.051 242	167.53	2573.5	2406.0	0.572 40	8.2555	7.6831	1.007 89	19 515.
41	0.007 787 8	991.79	0.053 871	171.71	2575.3	2403.6	0.585 73	8.2368	7.6511	1.008 28	18 563.
42	0.008 209 6	991.40	0.056 614	175.89	2577.1	2401.2	0.599 01	8.2182	7.6192	1.008 68	17 664.
43	0.008 650 8	991.00	0.059 474	180.07	2578.9	2398.8	0.612 25	8.1998	7.5875	1.009 09	16 814.
44	0.009 112 4	990.59	0.062 457	184.25	2580.6	2396.4	0.625 45	8.1815	7.5560	1.009 50	16 011.
45	0.009 595 0	990.17	0.065 565	188.43	2582.4	2394.0	0.638 61	8.1633	7.5247	1.009 92	15 252.
46	0.010 099	989.75	0.068 803	192.62	2584.2	2391.6	0.651 73	8.1453	7.4936	1.010 36	14 534.
47	0.010 627	989.32	0.072 176	196.80	2586.0	2389.2	0.664 81	8.1275	7.4627	1.010 79	13 855.
48	0.011 177	988.89	0.075 688	200.98	2587.8	2386.8	0.677 85	8.1098	7.4320	1.011 24	13 212.
49	0.011 752	988.44	0.079 343	205.16	2589.5	2384.4	0.690 85	8.0922	7.4014	1.011 69	12 603.
50	0.012 352	988.00	0.083 147	209.34	2591.3	2381.9	0.703 81	8.0748	7.3710	1.012 15	12 027.
51	0.012 978	987.54	0.087 103	213.52	2593.1	2379.5	0.716 73	8.0576	7.3408	1.012 62	11 481.
52	0.013 631	987.08	0.091 217	217.71	2594.8	2377.1	0.729 61	8.0404	7.3108	1.013 09	10 963.
53	0.014 312	986.61	0.095 494	221.89	2596.6	2374.7	0.742 45	8.0234	7.2810	1.013 57	10 472.
54	0.015 022	986.14	0.099 938	226.07	2598.3	2372.3	0.755 26	8.0066	7.2513	1.014 06	10 006.

Table 1. Saturation (Temperature) (continued)

		Density, kg/m³		Enthalpy, kJ/kg			Entropy, kJ/(kg·K)			Volume, cm³/g	
t, °C	p, MPa	ρ_L	ρ_V	h_L	h_V	Δh	s_L	s_V	Δs	v_L	v_V
55	0.015 762	985.66	0.104 56	230.26	2600.1	2369.8	0.768 02	7.9898	7.2218	1.014 55	9564.3
56	0.016 533	985.17	0.109 35	234.44	2601.8	2367.4	0.780 75	7.9732	7.1925	1.015 05	9144.8
57	0.017 336	984.68	0.114 33	238.62	2603.6	2365.0	0.793 44	7.9568	7.1633	1.015 56	8746.6
58	0.018 171	984.18	0.119 50	242.81	2605.3	2362.5	0.806 10	7.9404	7.1343	1.016 08	8368.3
59	0.019 041	983.67	0.124 86	246.99	2607.1	2360.1	0.818 71	7.9242	7.1055	1.016 60	8008.9
60	0.019 946	983.16	0.130 43	251.18	2608.8	2357.7	0.831 29	7.9081	7.0769	1.017 13	7667.2
61	0.020 888	982.64	0.136 20	255.37	2610.6	2355.2	0.843 84	7.8922	7.0484	1.017 66	7342.4
62	0.021 867	982.12	0.142 18	259.55	2612.3	2352.8	0.856 34	7.8764	7.0200	1.018 21	7033.5
63	0.022 885	981.59	0.148 38	263.74	2614.0	2350.3	0.868 82	7.8607	6.9918	1.018 75	6739.6
64	0.023 943	981.06	0.154 80	267.93	2615.8	2347.8	0.881 25	7.8451	6.9638	1.019 31	6459.8
65	0.025 042	980.52	0.161 46	272.12	2617.5	2345.4	0.893 65	7.8296	6.9359	1.019 87	6193.5
66	0.026 183	979.97	0.168 35	276.30	2619.2	2342.9	0.906 02	7.8142	6.9082	1.020 44	5939.9
67	0.027 368	979.42	0.175 49	280.49	2621.0	2340.5	0.918 35	7.7990	6.8807	1.021 01	5698.4
68	0.028 599	978.86	0.182 88	284.68	2622.7	2338.0	0.930 64	7.7839	6.8532	1.021 59	5468.2
69	0.029 876	978.30	0.190 52	288.87	2624.4	2335.5	0.942 91	7.7689	6.8260	1.022 18	5248.8
70	0.031 201	977.73	0.198 43	293.07	2626.1	2333.0	0.955 13	7.7540	6.7989	1.022 77	5039.5
71	0.032 575	977.16	0.206 61	297.26	2627.8	2330.5	0.967 33	7.7392	6.7719	1.023 37	4840.0
72	0.034 000	976.58	0.215 07	301.45	2629.5	2328.1	0.979 49	7.7246	6.7451	1.023 98	4649.6
73	0.035 478	976.00	0.223 82	305.64	2631.2	2325.6	0.991 61	7.7100	6.7184	1.024 59	4468.0
74	0.037 009	975.41	0.232 85	309.84	2632.9	2323.1	1.0037	7.6955	6.6918	1.025 21	4294.5
75	0.038 595	974.81	0.242 19	314.03	2634.6	2320.6	1.0158	7.6812	6.6654	1.025 84	4128.9
76	0.040 239	974.22	0.251 84	318.22	2636.3	2318.1	1.0278	7.6670	6.6392	1.026 47	3970.8
77	0.041 941	973.61	0.261 80	322.42	2638.0	2315.6	1.0398	7.6528	6.6130	1.027 10	3819.7
78	0.043 703	973.00	0.272 09	326.62	2639.7	2313.0	1.0517	7.6388	6.5871	1.027 75	3675.2
79	0.045 527	972.39	0.282 71	330.81	2641.3	2310.5	1.0637	7.6249	6.5612	1.028 40	3537.2
80	0.047 414	971.77	0.293 67	335.01	2643.0	2308.0	1.0756	7.6111	6.5355	1.029 05	3405.2
81	0.049 367	971.14	0.304 98	339.21	2644.7	2305.5	1.0874	7.5973	6.5099	1.029 72	3278.9
82	0.051 387	970.51	0.316 65	343.41	2646.4	2302.9	1.0993	7.5837	6.4844	1.030 38	3158.1
83	0.053 476	969.88	0.328 68	347.61	2648.0	2300.4	1.1111	7.5702	6.4591	1.031 06	3042.5
84	0.055 635	969.24	0.341 09	351.81	2649.7	2297.9	1.1229	7.5567	6.4339	1.031 74	2931.8
85	0.057 867	968.59	0.353 88	356.01	2651.3	2295.3	1.1346	7.5434	6.4088	1.032 43	2825.8
86	0.060 173	967.94	0.367 06	360.22	2653.0	2292.8	1.1463	7.5302	6.3838	1.033 12	2724.4
87	0.062 556	967.29	0.380 64	364.42	2654.6	2290.2	1.1580	7.5170	6.3590	1.033 82	2627.1
88	0.065 017	966.63	0.394 64	368.63	2656.3	2287.6	1.1696	7.5040	6.3343	1.034 52	2534.0
89	0.067 558	965.96	0.409 05	372.83	2657.9	2285.1	1.1813	7.4910	6.3097	1.035 24	2444.7
90	0.070 182	965.30	0.423 90	377.04	2659.5	2282.5	1.1929	7.4781	6.2853	1.035 95	2359.1
91	0.072 890	964.62	0.439 18	381.25	2661.2	2279.9	1.2044	7.4653	6.2609	1.036 68	2277.0
92	0.075 684	963.94	0.454 91	385.46	2662.8	2277.3	1.2160	7.4526	6.2367	1.037 41	2198.2
93	0.078 568	963.26	0.471 11	389.67	2664.4	2274.7	1.2275	7.4400	6.2126	1.038 14	2122.7
94	0.081 541	962.57	0.487 77	393.88	2666.0	2272.1	1.2389	7.4275	6.1886	1.038 88	2050.2
95	0.084 608	961.88	0.504 91	398.09	2667.6	2269.5	1.2504	7.4151	6.1647	1.039 63	1980.6
96	0.087 771	961.18	0.522 54	402.30	2669.2	2266.9	1.2618	7.4027	6.1409	1.040 38	1913.7
97	0.091 030	960.48	0.540 67	406.52	2670.8	2264.3	1.2732	7.3904	6.1172	1.041 14	1849.6
98	0.094 390	959.78	0.559 31	410.73	2672.4	2261.7	1.2846	7.3783	6.0937	1.041 91	1787.9
99	0.097 852	959.06	0.578 47	414.95	2674.0	2259.0	1.2959	7.3661	6.0702	1.042 68	1728.7
100	0.101 42	958.35	0.598 17	419.17	2675.6	2256.4	1.3072	7.3541	6.0469	1.043 46	1671.8
101	0.105 09	957.63	0.618 41	423.39	2677.1	2253.8	1.3185	7.3422	6.0237	1.044 25	1617.1
102	0.108 87	956.90	0.639 20	427.61	2678.7	2251.1	1.3297	7.3303	6.0006	1.045 04	1564.4
103	0.112 77	956.18	0.660 56	431.83	2680.3	2248.5	1.3410	7.3185	5.9775	1.045 83	1513.9
104	0.116 78	955.44	0.682 50	436.05	2681.8	2245.8	1.3522	7.3068	5.9546	1.046 64	1465.2
105	0.120 90	954.70	0.705 03	440.27	2683.4	2243.1	1.3633	7.2952	5.9318	1.047 44	1418.4
106	0.125 15	953.96	0.728 16	444.50	2684.9	2240.4	1.3745	7.2836	5.9091	1.048 26	1373.3
107	0.129 52	953.22	0.751 90	448.73	2686.5	2237.7	1.3856	7.2721	5.8865	1.049 08	1330.0
108	0.134 01	952.46	0.776 27	452.95	2688.0	2235.1	1.3967	7.2607	5.8640	1.049 91	1288.2
109	0.138 63	951.71	0.801 27	457.18	2689.5	2232.4	1.4078	7.2493	5.8416	1.050 74	1248.0

Table 1. Saturation (Temperature) (continued)

		Density, kg/m³		Enthalpy, kJ/kg			Entropy, kJ/(kg·K)			Volume, cm³/g	
t, °C	p, MPa	ρ_L	ρ_V	h_L	h_V	Δh	s_L	s_V	Δs	v_L	v_V
110	0.143 38	950.95	0.826 93	461.42	2691.1	2229.6	1.4188	7.2381	5.8193	1.051 58	1209.3
111	0.148 26	950.18	0.853 25	465.65	2692.6	2226.9	1.4298	7.2269	5.7970	1.052 43	1172.0
112	0.153 28	949.41	0.880 24	469.88	2694.1	2224.2	1.4408	7.2157	5.7749	1.053 28	1136.1
113	0.158 44	948.64	0.907 92	474.12	2695.6	2221.5	1.4518	7.2047	5.7529	1.054 14	1101.4
114	0.163 74	947.86	0.936 30	478.35	2697.1	2218.7	1.4628	7.1937	5.7309	1.055 00	1068.0
115	0.169 18	947.08	0.965 40	482.59	2698.6	2216.0	1.4737	7.1828	5.7091	1.055 88	1035.8
116	0.174 77	946.30	0.995 22	486.83	2700.1	2213.2	1.4846	7.1719	5.6873	1.056 75	1004.8
117	0.180 52	945.50	1.0258	491.08	2701.5	2210.5	1.4954	7.1611	5.6657	1.057 64	974.86
118	0.186 41	944.71	1.0571	495.32	2703.0	2207.7	1.5063	7.1504	5.6441	1.058 53	945.98
119	0.192 46	943.91	1.0892	499.56	2704.5	2204.9	1.5171	7.1397	5.6226	1.059 42	918.11
120	0.198 67	943.11	1.1221	503.81	2705.9	2202.1	1.5279	7.1291	5.6012	1.060 33	891.21
121	0.205 05	942.30	1.1557	508.06	2707.4	2199.3	1.5387	7.1186	5.5799	1.061 23	865.25
122	0.211 59	941.49	1.1902	512.31	2708.8	2196.5	1.5494	7.1081	5.5587	1.062 15	840.19
123	0.218 30	940.67	1.2255	516.56	2710.3	2193.7	1.5602	7.0977	5.5375	1.063 07	815.98
124	0.225 18	939.85	1.2617	520.82	2711.7	2190.9	1.5709	7.0873	5.5165	1.064 00	792.61
125	0.232 24	939.02	1.2987	525.07	2713.1	2188.0	1.5816	7.0770	5.4955	1.064 94	770.03
126	0.239 47	938.19	1.3365	529.33	2714.5	2185.2	1.5922	7.0668	5.4746	1.065 88	748.21
127	0.246 89	937.36	1.3753	533.59	2715.9	2182.3	1.6029	7.0566	5.4538	1.066 83	727.13
128	0.254 50	936.52	1.4149	537.85	2717.3	2179.5	1.6135	7.0465	5.4330	1.067 78	706.75
129	0.262 29	935.68	1.4555	542.12	2718.7	2176.6	1.6241	7.0364	5.4124	1.068 74	687.05
130	0.270 28	934.83	1.4970	546.38	2720.1	2173.7	1.6346	7.0264	5.3918	1.069 71	668.00
131	0.278 46	933.98	1.5394	550.65	2721.5	2170.8	1.6452	7.0165	5.3713	1.070 68	649.59
132	0.286 85	933.13	1.5828	554.92	2722.8	2167.9	1.6557	7.0066	5.3509	1.071 66	631.77
133	0.295 43	932.27	1.6272	559.19	2724.2	2165.0	1.6662	6.9967	5.3305	1.072 65	614.54
134	0.304 23	931.41	1.6726	563.47	2725.5	2162.1	1.6767	6.9869	5.3102	1.073 65	597.86
135	0.313 23	930.54	1.7190	567.74	2726.9	2159.1	1.6872	6.9772	5.2900	1.074 65	581.73
136	0.322 45	929.67	1.7664	572.02	2728.2	2156.2	1.6976	6.9675	5.2699	1.075 66	566.11
137	0.331 88	928.79	1.8149	576.30	2729.5	2153.2	1.7081	6.9579	5.2498	1.076 67	550.99
138	0.341 54	927.91	1.8644	580.59	2730.8	2150.3	1.7185	6.9483	5.2298	1.077 69	536.36
139	0.351 43	927.02	1.9150	584.87	2732.1	2147.3	1.7289	6.9388	5.2099	1.078 72	522.18
140	0.361 54	926.13	1.9667	589.16	2733.4	2144.3	1.7392	6.9293	5.1901	1.079 76	508.45
141	0.371 89	925.24	2.0196	593.45	2734.7	2141.3	1.7496	6.9199	5.1703	1.080 80	495.16
142	0.382 47	924.34	2.0735	597.74	2736.0	2138.3	1.7599	6.9105	5.1506	1.081 85	482.27
143	0.393 29	923.44	2.1286	602.04	2737.3	2135.2	1.7702	6.9011	5.1309	1.082 91	469.79
144	0.404 37	922.54	2.1849	606.34	2738.5	2132.2	1.7805	6.8919	5.1114	1.083 97	457.69
145	0.415 68	921.62	2.2423	610.64	2739.8	2129.2	1.7907	6.8826	5.0919	1.085 04	445.96
146	0.427 26	920.71	2.3010	614.94	2741.0	2126.1	1.8010	6.8734	5.0724	1.086 12	434.59
147	0.439 09	919.79	2.3609	619.25	2742.3	2123.0	1.8112	6.8643	5.0530	1.087 20	423.57
148	0.451 18	918.87	2.4220	623.56	2743.5	2119.9	1.8214	6.8552	5.0337	1.088 30	412.88
149	0.463 54	917.94	2.4844	627.87	2744.7	2116.9	1.8316	6.8461	5.0145	1.089 40	402.51
150	0.476 16	917.01	2.5481	632.18	2745.9	2113.7	1.8418	6.8371	4.9953	1.090 50	392.45
151	0.489 07	916.07	2.6130	636.50	2747.1	2110.6	1.8520	6.8281	4.9761	1.091 62	382.69
152	0.502 25	915.13	2.6793	640.81	2748.3	2107.5	1.8621	6.8192	4.9571	1.092 74	373.23
153	0.515 71	914.19	2.7470	645.14	2749.5	2104.3	1.8722	6.8103	4.9380	1.093 87	364.04
154	0.529 46	913.24	2.8160	649.46	2750.7	2101.2	1.8823	6.8014	4.9191	1.095 01	355.12
155	0.543 50	912.28	2.8863	653.79	2751.8	2098.0	1.8924	6.7926	4.9002	1.096 15	346.46
156	0.557 84	911.33	2.9581	658.12	2753.0	2094.8	1.9025	6.7838	4.8814	1.097 30	338.05
157	0.572 47	910.36	3.0313	662.45	2754.1	2091.6	1.9125	6.7751	4.8626	1.098 46	329.89
158	0.587 42	909.40	3.1059	666.79	2755.2	2088.4	1.9225	6.7664	4.8439	1.099 63	321.96
159	0.602 67	908.42	3.1821	671.13	2756.3	2085.2	1.9326	6.7578	4.8252	1.100 81	314.26
160	0.618 23	907.45	3.2596	675.47	2757.4	2082.0	1.9426	6.7491	4.8066	1.101 99	306.78
161	0.634 12	906.47	3.3387	679.82	2758.5	2078.7	1.9525	6.7406	4.7880	1.103 18	299.51
162	0.650 33	905.49	3.4194	684.17	2759.6	2075.5	1.9625	6.7320	4.7695	1.104 38	292.45
163	0.666 86	904.50	3.5016	688.52	2760.7	2072.2	1.9725	6.7235	4.7511	1.105 59	285.59
164	0.683 73	903.50	3.5853	692.88	2761.8	2068.9	1.9824	6.7150	4.7327	1.106 80	278.92

Table 1. Saturation (Temperature) (continued)

		Density, kg/m³		Enthalpy, kJ/kg			Entropy, kJ/(kg·K)			Volume, cm³/g	
t, °C	p, MPa	ρ_L	ρ_V	h_L	h_V	Δh	s_L	s_V	Δs	v_L	v_V
165	0.700 93	902.51	3.6707	697.24	2762.8	2065.6	1.9923	6.7066	4.7143	1.108 03	272.43
166	0.718 48	901.50	3.7576	701.60	2763.9	2062.3	2.0022	6.6982	4.6960	1.109 26	266.12
167	0.736 38	900.50	3.8462	705.96	2764.9	2058.9	2.0121	6.6898	4.6778	1.110 50	259.99
168	0.754 62	899.49	3.9365	710.33	2765.9	2055.6	2.0220	6.6815	4.6596	1.111 75	254.03
169	0.773 22	898.47	4.0285	714.71	2766.9	2052.2	2.0318	6.6732	4.6414	1.113 00	248.23
170	0.792 19	897.45	4.1222	719.08	2767.9	2048.8	2.0417	6.6650	4.6233	1.114 27	242.59
171	0.811 52	896.43	4.2176	723.46	2768.9	2045.4	2.0515	6.6567	4.6053	1.115 54	237.10
172	0.831 22	895.40	4.3148	727.85	2769.9	2042.0	2.0613	6.6485	4.5872	1.116 82	231.76
173	0.851 30	894.36	4.4138	732.23	2770.8	2038.6	2.0711	6.6404	4.5693	1.118 11	226.56
174	0.871 76	893.33	4.5146	736.63	2771.8	2035.1	2.0809	6.6322	4.5514	1.119 41	221.50
175	0.892 60	892.28	4.6172	741.02	2772.7	2031.7	2.0906	6.6241	4.5335	1.120 72	216.58
176	0.913 84	891.24	4.7217	745.42	2773.6	2028.2	2.1004	6.6161	4.5157	1.122 04	211.79
177	0.935 47	890.18	4.8281	749.82	2774.5	2024.7	2.1101	6.6080	4.4979	1.123 36	207.12
178	0.957 51	889.13	4.9364	754.23	2775.4	2021.2	2.1198	6.6000	4.4802	1.124 70	202.58
179	0.979 95	888.07	5.0466	758.64	2776.3	2017.7	2.1296	6.5920	4.4625	1.126 04	198.15
180	1.0028	887.00	5.1588	763.05	2777.2	2014.2	2.1392	6.5840	4.4448	1.127 40	193.84
181	1.0261	885.93	5.2730	767.47	2778.1	2010.6	2.1489	6.5761	4.4272	1.128 76	189.64
182	1.0498	884.85	5.3893	771.90	2778.9	2007.0	2.1586	6.5682	4.4096	1.130 13	185.55
183	1.0739	883.77	5.5076	776.32	2779.8	2003.4	2.1683	6.5603	4.3921	1.131 51	181.57
184	1.0985	882.69	5.6279	780.75	2780.6	1999.8	2.1779	6.5525	4.3746	1.132 90	177.69
185	1.1235	881.60	5.7504	785.19	2781.4	1996.2	2.1875	6.5447	4.3571	1.134 30	173.90
186	1.1489	880.50	5.8750	789.63	2782.2	1992.6	2.1971	6.5369	4.3397	1.135 71	170.21
187	1.1748	879.40	6.0018	794.07	2783.0	1988.9	2.2067	6.5291	4.3223	1.137 13	166.62
188	1.2011	878.30	6.1308	798.52	2783.8	1985.3	2.2163	6.5213	4.3050	1.138 56	163.11
189	1.2280	877.19	6.2620	802.97	2784.5	1981.6	2.2259	6.5136	4.2877	1.140 00	159.69
190	1.2552	876.08	6.3954	807.43	2785.3	1977.9	2.2355	6.5059	4.2704	1.141 45	156.36
191	1.2830	874.96	6.5312	811.89	2786.0	1974.1	2.2450	6.4982	4.2532	1.142 91	153.11
192	1.3112	873.83	6.6692	816.36	2786.7	1970.4	2.2546	6.4906	4.2360	1.144 38	149.94
193	1.3399	872.70	6.8096	820.83	2787.4	1966.6	2.2641	6.4830	4.2188	1.145 86	146.85
194	1.3691	871.57	6.9524	825.31	2788.1	1962.8	2.2736	6.4754	4.2017	1.147 36	143.83
195	1.3988	870.43	7.0976	829.79	2788.8	1959.0	2.2832	6.4678	4.1846	1.148 86	140.89
196	1.4290	869.29	7.2453	834.28	2789.5	1955.2	2.2926	6.4602	4.1676	1.150 37	138.02
197	1.4597	868.14	7.3954	838.77	2790.1	1951.4	2.3021	6.4527	4.1505	1.151 89	135.22
198	1.4909	866.98	7.5480	843.26	2790.8	1947.5	2.3116	6.4451	4.1335	1.153 43	132.48
199	1.5227	865.82	7.7032	847.76	2791.4	1943.6	2.3211	6.4376	4.1166	1.154 97	129.82
200	1.5549	864.66	7.8610	852.27	2792.0	1939.7	2.3305	6.4302	4.0996	1.156 53	127.21
201	1.5877	863.49	8.0214	856.78	2792.6	1935.8	2.3400	6.4227	4.0827	1.158 09	124.67
202	1.6210	862.31	8.1844	861.30	2793.2	1931.9	2.3494	6.4152	4.0658	1.159 67	122.18
203	1.6549	861.13	8.3501	865.82	2793.7	1927.9	2.3588	6.4078	4.0490	1.161 26	119.76
204	1.6893	859.95	8.5186	870.35	2794.3	1923.9	2.3683	6.4004	4.0322	1.162 86	117.39
205	1.7243	858.76	8.6898	874.88	2794.8	1919.9	2.3777	6.3930	4.0154	1.164 48	115.08
206	1.7598	857.56	8.8638	879.42	2795.3	1915.9	2.3871	6.3856	3.9986	1.166 10	112.82
207	1.7959	856.36	9.0406	883.96	2795.9	1911.9	2.3964	6.3783	3.9819	1.167 74	110.61
208	1.8326	855.15	9.2203	888.51	2796.3	1907.8	2.4058	6.3710	3.9651	1.169 39	108.46
209	1.8698	853.94	9.4029	893.07	2796.8	1903.7	2.4152	6.3636	3.9484	1.171 05	106.35
210	1.9077	852.72	9.5885	897.63	2797.3	1899.6	2.4245	6.3563	3.9318	1.172 72	104.29
211	1.9461	851.49	9.7770	902.20	2797.7	1895.5	2.4339	6.3490	3.9151	1.174 41	102.28
212	1.9851	850.26	9.9686	906.77	2798.1	1891.4	2.4432	6.3417	3.8985	1.176 11	100.31
213	2.0247	849.03	10.163	911.35	2798.5	1887.2	2.4526	6.3345	3.8819	1.177 82	98.394
214	2.0650	847.79	10.361	915.94	2798.9	1883.0	2.4619	6.3272	3.8653	1.179 54	96.516
215	2.1058	846.54	10.562	920.53	2799.3	1878.8	2.4712	6.3200	3.8488	1.181 28	94.679
216	2.1473	845.29	10.766	925.12	2799.7	1874.6	2.4805	6.3128	3.8323	1.183 03	92.884
217	2.1894	844.03	10.973	929.73	2800.0	1870.3	2.4898	6.3056	3.8158	1.184 79	91.129
218	2.2322	842.77	11.184	934.34	2800.3	1866.0	2.4991	6.2984	3.7993	1.186 57	89.413
219	2.2756	841.50	11.398	938.96	2800.7	1861.7	2.5084	6.2912	3.7828	1.188 36	87.734

Table 1. Saturation (Temperature) (continued)

		Density, kg/m³		Enthalpy, kJ/kg			Entropy, kJ/(kg·K)			Volume, cm³/g	
t, °C	p, MPa	ρ_L	ρ_V	h_L	h_V	Δh	s_L	s_V	Δs	v_L	v_V
220	2.3196	840.22	11.615	943.58	2800.9	1857.4	2.5177	6.2840	3.7663	1.190 17	86.092
221	2.3643	838.94	11.836	948.21	2801.2	1853.0	2.5269	6.2768	3.7499	1.191 98	84.486
222	2.4096	837.65	12.060	952.85	2801.5	1848.6	2.5362	6.2697	3.7335	1.193 82	82.916
223	2.4556	836.35	12.288	957.49	2801.7	1844.2	2.5455	6.2625	3.7171	1.195 67	81.379
224	2.5023	835.05	12.520	962.14	2801.9	1839.8	2.5547	6.2554	3.7007	1.197 53	79.875
225	2.5497	833.75	12.755	966 80	2802.1	1835.4	2.5640	6 2483	3.6843	1.199 40	78.403
226	2.5978	832.43	12.993	971.46	2802.3	1830.9	2.5732	6.2412	3.6680	1.201 30	76.964
227	2.6466	831.12	13.235	976.13	2802.5	1826.4	2.5824	6.2341	3.6516	1.203 20	75.554
228	2.6960	829.79	13.482	980.81	2802.7	1821.8	2.5917	6.2270	3.6353	1.205 12	74.175
229	2.7462	828.46	13.732	985.50	2802.8	1817.3	2.6009	6.2199	3.6190	1.207 06	72.825
230	2.7971	827.12	13.985	990.19	2802.9	1812.7	2.6101	6.2128	3.6027	1.209 02	71.503
231	2.8487	825.77	14.243	994.89	2803.0	1808.1	2.6193	6.2057	3.5864	1.210 98	70.210
232	2.9010	824.42	14.505	999.60	2803.1	1803.5	2.6285	6.1987	3.5702	1.212 97	68.943
233	2.9541	823.06	14.771	1004.3	2803.1	1798.8	2.6377	6.1916	3.5539	1.214 97	67.702
234	3.0080	821.70	15.040	1009.0	2803.2	1794.1	2.6469	6.1845	3.5376	1.216 99	66.488
235	3.0625	820.33	15.314	1013.8	2803.2	1789.4	2.6561	6.1775	3.5214	1.219 02	65.298
236	3.1179	818.95	15.593	1018.5	2803.2	1784.7	2.6653	6.1704	3.5052	1.221 08	64.133
237	3.1740	817.56	15.875	1023.3	2803.1	1779.9	2.6745	6.1634	3.4890	1.223 15	62.991
238	3.2308	816.17	16.162	1028.0	2803.1	1775.1	2.6836	6.1564	3.4727	1.225 23	61.873
239	3.2885	814.77	16.453	1032.8	2803.0	1770.3	2.6928	6.1493	3.4565	1.227 34	60.778
240	3.3469	813.37	16.749	1037.6	2803.0	1765.4	2.7020	6.1423	3.4403	1.229 46	59.705
241	3.4062	811.95	17.049	1042.3	2802.9	1760.5	2.7111	6.1353	3.4241	1.231 60	58.654
242	3.4662	810.53	17.354	1047.1	2802.7	1755.6	2.7203	6.1282	3.4079	1.233 76	57.623
243	3.5270	809.10	17.664	1051.9	2802.6	1750.7	2.7295	6.1212	3.3918	1.235 94	56.613
244	3.5887	807.67	17.978	1056.7	2802.4	1745.7	2.7386	6.1142	3.3756	1.238 13	55.624
245	3.6512	806.22	18.297	1061.5	2802.2	1740.7	2.7478	6.1072	3.3594	1.240 35	54.654
246	3.7145	804.77	18.621	1066.4	2802.0	1735.6	2.7569	6.1002	3.3432	1.242 59	53.703
247	3.7786	803.32	18.950	1071.2	2801.8	1730.6	2.7661	6.0931	3.3270	1.244 84	52.771
248	3.8436	801.85	19.284	1076.1	2801.5	1725.5	2.7752	6.0861	3.3109	1.247 12	51.857
249	3.9095	800.38	19.623	1080.9	2801.2	1720.3	2.7844	6.0791	3.2947	1.249 41	50.961
250	3.9762	798.89	19.967	1085.8	2800.9	1715.2	2.7935	6.0721	3.2785	1.251 73	50.083
251	4.0438	797.40	20.316	1090.6	2800.6	1710.0	2.8027	6.0650	3.2624	1.254 07	49.222
252	4.1122	795.91	20.671	1095.5	2800.3	1704.7	2.8118	6.0580	3.2462	1.256 43	48.377
253	4.1815	794.40	21.031	1100.4	2799.9	1699.5	2.8210	6.0510	3.2300	1.258 81	47.548
254	4.2518	792.89	21.397	1105.3	2799.5	1694.2	2.8301	6.0439	3.2138	1.261 21	46.736
255	4.3229	791.37	21.768	1110.2	2799.1	1688.8	2.8392	6.0369	3.1977	1.263 64	45.938
256	4.3949	789.83	22.145	1115.2	2798.6	1683.5	2.8484	6.0298	3.1815	1.266 09	45.156
257	4.4679	788.30	22.528	1120.1	2798.2	1678.1	2.8575	6.0228	3.1653	1.268 56	44.389
258	4.5417	786.75	22.917	1125.0	2797.7	1672.6	2.8667	6.0157	3.1491	1.271 06	43.637
259	4.6165	785.19	23.311	1130.0	2797.1	1667.2	2.8758	6.0087	3.1329	1.273 58	42.898
260	4.6923	783.63	23.712	1135.0	2796.6	1661.6	2.8849	6.0016	3.1167	1.276 12	42.173
261	4.7689	782.05	24.118	1139.9	2796.0	1656.1	2.8941	5.9945	3.1004	1.278 69	41.462
262	4.8466	780.47	24.531	1144.9	2795.4	1650.5	2.9032	5.9874	3.0842	1.281 28	40.764
263	4.9252	778.88	24.951	1149.9	2794.8	1644.9	2.9124	5.9804	3.0680	1.283 90	40.079
264	5.0047	777.27	25.377	1154.9	2794.2	1639.2	2.9215	5.9732	3.0517	1.286 55	39.406
265	5.0853	775.66	25.809	1160.0	2793.5	1633.5	2.9307	5.9661	3.0354	1.289 22	38.746
266	5.1668	774.04	26.248	1165.0	2792.8	1627.8	2.9398	5.9590	3.0192	1.291 92	38.098
267	5.2494	772.41	26.694	1170.0	2792.1	1622.0	2.9490	5.9519	3.0029	1.294 65	37.462
268	5.3329	770.77	27.147	1175.1	2791.3	1616.2	2.9582	5.9447	2.9866	1.297 40	36.837
269	5.4174	769.12	27.606	1180.2	2790.5	1610.3	2.9673	5.9376	2.9703	1.300 19	36.223
270	5.5030	767.46	28.073	1185.3	2789.7	1604.4	2.9765	5.9304	2.9539	1.303 00	35.621
271	5.5896	765.79	28.548	1190.4	2788.8	1598.5	2.9857	5.9232	2.9376	1.305 84	35.029
272	5.6772	764.11	29.029	1195.5	2788.0	1592.5	2.9948	5.9160	2.9212	1.308 71	34.448
273	5.7659	762.42	29.518	1200.6	2787.1	1586.5	3.0040	5.9088	2.9048	1.311 61	33.877
274	5.8556	760.72	30.015	1205.7	2786.1	1580.4	3.0132	5.9016	2.8884	1.314 55	33.317

Table 1. Saturation (Temperature) (continued)

		Density, kg/m³		Enthalpy, kJ/kg			Entropy, kJ/(kg·K)			Volume, cm³/g	
t, °C	p, MPa	ρ_L	ρ_V	h_L	h_V	Δh	s_L	s_V	Δs	v_L	v_V
275	5.9464	759.00	30.520	1210.9	2785.2	1574.3	3.0224	5.8944	2.8720	1.317 51	32.766
276	6.0383	757.28	31.032	1216.1	2784.2	1568.1	3.0316	5.8871	2.8555	1.320 51	32.225
277	6.1312	755.55	31.553	1221.3	2783.1	1561.9	3.0408	5.8798	2.8390	1.323 54	31.693
278	6.2252	753.80	32.082	1226.4	2782.1	1555.6	3.0500	5.8725	2.8225	1.326 61	31.171
279	6.3203	752.04	32.619	1231.7	2781.0	1549.3	3.0592	5.8652	2.8060	1.329 71	30.657
280	6.4166	750.28	33.165	1236.9	2779.9	1543.0	3.0685	5.8579	2.7894	1.332 84	30.153
281	6.5139	748.49	33.719	1242.1	2778.7	1536.6	3.0777	5.8506	2.7729	1.336 02	29.657
282	6.6124	746.70	34.283	1247.4	2777.5	1530.1	3.0869	5.8432	2.7563	1.339 22	29.169
283	6.7120	744.90	34.855	1252.7	2776.3	1523.6	3.0962	5.8358	2.7396	1.342 47	28.690
284	6.8128	743.08	35.437	1257.9	2775.0	1517.1	3.1054	5.8284	2.7229	1.345 75	28.219
285	6.9147	741.25	36.028	1263.2	2773.7	1510.5	3.1147	5.8209	2.7062	1.349 07	27.756
286	7.0177	739.41	36.629	1268.6	2772.4	1503.8	3.1240	5.8135	2.6895	1.352 43	27.301
287	7.1220	737.55	37.239	1273.9	2771.0	1497.1	3.1333	5.8060	2.6727	1.355 84	26.853
288	7.2274	735.68	37.860	1279.3	2769.6	1490.4	3.1426	5.7985	2.6559	1.359 28	26.413
289	7.3340	733.80	38.490	1284.6	2768.2	1483.5	3.1519	5.7909	2.6390	1.362 77	25.981
290	7.4418	731.91	39.132	1290.0	2766.7	1476.7	3.1612	5.7834	2.6222	1.366 30	25.555
291	7.5508	730.00	39.783	1295.4	2765.2	1469.7	3.1705	5.7758	2.6052	1.369 87	25.136
292	7.6610	728.07	40.446	1300.9	2763.6	1462.7	3.1799	5.7681	2.5883	1.373 49	24.724
293	7.7725	726.13	41.120	1306.3	2762.0	1455.7	3.1892	5.7605	2.5712	1.377 16	24.319
294	7.8852	724.18	41.805	1311.8	2760.4	1448.6	3.1986	5.7528	2.5542	1.380 87	23.921
295	7.9991	722.21	42.501	1317.3	2758.7	1441.4	3.2080	5.7451	2.5371	1.384 64	23.529
296	8.1143	720.23	43.210	1322.8	2757.0	1434.2	3.2174	5.7373	2.5199	1.388 45	23.143
297	8.2308	718.23	43.931	1328.3	2755.2	1426.9	3.2268	5.7295	2.5027	1.392 31	22.763
298	8.3485	716.21	44.664	1333.8	2753.4	1419.5	3.2362	5.7217	2.4854	1.396 23	22.390
299	8.4676	714.18	45.409	1339.4	2751.5	1412.1	3.2457	5.7138	2.4681	1.400 20	22.022
300	8.5879	712.14	46.168	1345.0	2749.6	1404.6	3.2552	5.7059	2.4507	1.404 23	21.660
301	8.7095	710.07	46.940	1350.6	2747.7	1397.1	3.2647	5.6979	2.4333	1.408 31	21.304
302	8.8325	707.99	47.725	1356.3	2745.7	1389.4	3.2742	5.6899	2.4158	1.412 45	20.953
303	8.9568	705.89	48.525	1361.9	2743.7	1381.7	3.2837	5.6819	2.3982	1.416 65	20.608
304	9.0824	703.77	49.338	1367.6	2741.6	1374.0	3.2932	5.6738	2.3806	1.420 91	20.268
305	9.2094	701.64	50.167	1373.3	2739.4	1366.1	3.3028	5.6657	2.3629	1.425 24	19.933
306	9.3378	699.48	51.010	1379.0	2737.2	1358.2	3.3124	5.6575	2.3452	1.429 63	19.604
307	9.4675	697.31	51.869	1384.8	2735.0	1350.2	3.3220	5.6493	2.3273	1.434 08	19.279
308	9.5986	695.12	52.743	1390.6	2732.7	1342.1	3.3316	5.6411	2.3094	1.438 61	18.960
309	9.7311	692.90	53.634	1396.4	2730.4	1334.0	3.3413	5.6327	2.2915	1.443 20	18.645
310	9.8651	690.67	54.541	1402.2	2727.9	1325.7	3.3510	5.6244	2.2734	1.447 87	18.335
311	10.000	688.42	55.466	1408.1	2725.5	1317.4	3.3607	5.6159	2.2553	1.452 61	18.029
312	10.137	686.14	56.408	1414.0	2723.0	1309.0	3.3704	5.6074	2.2370	1.457 43	17.728
313	10.275	683.84	57.368	1419.9	2720.4	1300.5	3.3802	5.5989	2.2187	1.462 32	17.431
314	10.415	681.52	58.346	1425.8	2717.8	1291.9	3.3900	5.5903	2.2003	1.467 30	17.139
315	10.556	679.18	59.344	1431.8	2715.1	1283.2	3.3998	5.5816	2.1818	1.472 36	16.851
316	10.699	676.81	60.361	1437.8	2712.3	1274.5	3.4097	5.5729	2.1632	1.477 51	16.567
317	10.843	674.42	61.398	1443.9	2709.5	1265.6	3.4195	5.5641	2.1445	1.482 75	16.287
318	10.989	672.00	62.457	1450.0	2706.6	1256.6	3.4295	5.5552	2.1257	1.488 09	16.011
319	11.136	669.56	63.537	1456.1	2703.6	1247.5	3.4394	5.5462	2.1068	1.493 51	15.739
320	11.284	667.09	64.638	1462.2	2700.6	1238.4	3.4494	5.5372	2.0878	1.499 04	15.471
321	11.434	664.60	65.763	1468.4	2697.5	1229.1	3.4595	5.5281	2.0686	1.504 67	15.206
322	11.586	662.07	66.912	1474.6	2694.3	1219.7	3.4695	5.5189	2.0494	1.510 40	14.945
323	11.740	659.52	68.084	1480.9	2691.1	1210.2	3.4797	5.5096	2.0300	1.516 25	14.688
324	11.895	656.94	69.282	1487.2	2687.7	1200.6	3.4898	5.5003	2.0105	1.522 21	14.434
325	12.051	654.33	70.506	1493.5	2684.3	1190.8	3.5000	5.4908	1.9908	1.528 29	14.183
326	12.209	651.68	71.757	1499.9	2680.8	1180.9	3.5103	5.4813	1.9710	1.534 49	13.936
327	12.369	649.01	73.036	1506.3	2677.3	1170.9	3.5206	5.4717	1.9511	1.540 81	13.692
328	12.530	646.30	74.344	1512.8	2673.6	1160.8	3.5309	5.4619	1.9310	1.547 27	13.451
329	12.693	643.55	75.682	1519.3	2669.9	1150.6	3.5413	5.4521	1.9108	1.553 87	13.213

Table 1. Saturation (Temperature) (continued)

		Density, kg/m³		Enthalpy, kJ/kg			Entropy, kJ/(kg·K)			Volume, cm³/g	
t, °C	p, MPa	ρ_L	ρ_V	h_L	h_V	Δh	s_L	s_V	Δs	v_L	v_V
330	12.858	640.77	77.050	1525.9	2666.0	1140.2	3.5518	5.4422	1.8903	1.560 61	12.979
331	13.024	637.96	78.452	1532.5	2662.1	1129.6	3.5623	5.4321	1.8698	1.567 51	12.747
332	13.193	635.10	79.887	1539.1	2658.1	1118.9	3.5729	5.4219	1.8490	1.574 56	12.518
333	13.362	632.20	81.356	1545.9	2653.9	1108.1	3.5835	5.4116	1.8281	1.581 77	12.292
334	13.534	629.27	82.863	1552.6	2649.7	1097.1	3.5943	5.4012	1.8069	1.589 15	12.068
335	13.707	626.29	84.407	1559.5	2645.4	1085.9	3.6050	5.3906	1.7856	1.596 71	11.847
336	13.882	623.26	85.991	1566.3	2640.9	1074.6	3.6159	5.3799	1.7640	1.604 47	11.629
337	14.059	620.19	87.616	1573.3	2636.3	1063.0	3.6268	5.3691	1.7422	1.612 41	11.413
338	14.238	617.07	89.284	1580.3	2631.6	1051.3	3.6378	5.3581	1.7202	1.620 57	11.200
339	14.418	613.89	90.998	1587.4	2626.8	1039.4	3.6489	5.3469	1.6980	1.628 95	10.989
340	14.601	610.67	92.759	1594.5	2621.8	1027.3	3.6601	5.3356	1.6755	1.637 55	10.781
341	14.785	607.38	94.570	1601.8	2616.8	1015.0	3.6714	5.3241	1.6527	1.646 40	10.574
342	14.971	604.04	96.433	1609.1	2611.5	1002.5	3.6828	5.3124	1.6296	1.655 51	10.370
343	15.159	600.64	98.351	1616.4	2606.1	989.7	3.6943	5.3005	1.6063	1.664 90	10.168
344	15.349	597.17	100.33	1623.9	2600.6	976.7	3.7059	5.2885	1.5826	1.674 57	9.9674
345	15.541	593.63	102.36	1631.5	2594.9	963.4	3.7176	5.2762	1.5586	1.684 56	9.7690
346	15.734	590.01	104.47	1639.1	2589.0	949.9	3.7295	5.2636	1.5342	1.694 88	9.5724
347	15.930	586.32	106.64	1646.9	2583.0	936.1	3.7414	5.2509	1.5094	1.705 56	9.3776
348	16.128	582.54	108.88	1654.8	2576.7	922.0	3.7536	5.2379	1.4843	1.716 62	9.1844
349	16.328	578.67	111.20	1662.8	2570.3	907.5	3.7659	5.2246	1.4587	1.728 10	8.9927
350	16.529	574.71	113.61	1670.9	2563.6	892.7	3.7784	5.2110	1.4326	1.740 02	8.8024
351	16.733	570.64	116.10	1679.1	2556.8	877.6	3.7910	5.1971	1.4061	1.752 43	8.6134
352	16.939	566.46	118.68	1687.5	2549.6	862.1	3.8039	5.1829	1.3790	1.765 36	8.4257
353	17.147	562.15	121.37	1696.1	2542.3	846.2	3.8170	5.1683	1.3514	1.778 88	8.2390
354	17.358	557.72	124.17	1704.8	2534.6	829.8	3.8303	5.1534	1.3231	1.793 02	8.0533
355	17.570	553.14	127.09	1713.7	2526.6	812.9	3.8439	5.1380	1.2942	1.807 86	7.8684
356	17.785	548.41	130.14	1722.8	2518.4	795.5	3.8577	5.1222	1.2645	1.823 47	7.6841
357	18.002	543.50	133.33	1732.2	2509.8	777.6	3.8719	5.1059	1.2340	1.839 93	7.5003
358	18.221	538.41	136.67	1741.7	2500.8	759.0	3.8864	5.0891	1.2026	1.857 33	7.3168
359	18.442	533.11	140.19	1751.5	2491.4	739.8	3.9014	5.0717	1.1703	1.875 78	7.1332
360	18.666	527.59	143.90	1761.7	2481.5	719.8	3.9167	5.0536	1.1369	1.895 41	6.9493
361	18.892	521.82	147.82	1772.1	2471.1	699.0	3.9325	5.0347	1.1023	1.916 35	6.7649
362	19.121	515.79	151.99	1782.9	2460.2	677.3	3.9488	5.0151	1.0663	1.938 79	6.5795
363	19.352	509.45	156.43	1794.1	2448.6	654.5	3.9656	4.9945	1.0288	1.962 90	6.3925
364	19.585	502.78	161.20	1805.7	2436.2	630.5	3.9831	4.9727	0.9896	1.988 94	6.2035
365	19.821	495.74	166.35	1817.8	2422.9	605.2	4.0014	4.9497	0.9483	2.0172	6.0115
366	20.060	488.27	171.95	1830.5	2408.7	578.2	4.0205	4.9251	0.9046	2.0480	5.8157
367	20.302	480.29	178.11	1843.8	2393.1	549.2	4.0406	4.8986	0.8580	2.0821	5.6145
368	20.546	471.67	184.98	1858.1	2375.9	517.8	4.0621	4.8697	0.8076	2.1201	5.4061
369	20.793	462.18	192.77	1873.5	2356.6	483.1	4.0853	4.8376	0.7523	2.1636	5.1875
370	21.044	451.43	201.84	1890.7	2334.5	443.8	4.1112	4.8012	0.6901	2.2152	4.9544
371	21.297	438.64	212.79	1910.6	2308.3	397.7	4.1412	4.7586	0.6175	2.2798	4.6995
372	21.554	422.26	226.84	1935.3	2275.5	340.3	4.1785	4.7059	0.5274	2.3682	4.4084
373	21.814	398.68	247.22	1969.7	2229.8	260.1	4.2308	4.6334	0.4026	2.5083	4.0450
t_c	22.064	322.00	322.00	2084.3	2084.3	0.	4.4070	4.4070	0.	3.1056	3.1056

(t_c = 373.946 °C)

APPENDIX 4: C# CODE FOR LATENT HEAT ALGORITHM

This appendix describes the components of the LatentHeat method. Section A4.3 contains the latent heat algorithm coded in the C# programming language.

A4.1 Latent Heat Algorithm

To organize the values used and calculated by the latent heat algorithm a data table, "Steam Boundary," is created with columns to hold the following data for each steam and liquid element pair:

Values Used

- Tracking code for decision tree end points
- Element numbers for liquid-steam boundary elements
- Node numbers for common nodes between liquid-steam boundary elements
- Temperatures for the liquid and steam elements
- Total fluid flux from the steam element
- Components (x and y/z) of fluid flux from the steam element
- Fluid density of the steam element

Values Calculated

- Saturation temperature at the liquid-steam boundary
- Enthalpy at the liquid-steam boundary
- Angle between x-axis and vector between common nodes for selected element pair geometries
- Length of liquid-steam boundary
- Magnitude of flux perpendicular or normal to the liquid-steam boundary

- Source/sink term
- Value of source/sink applied at the common nodes

A new flux array, lflx[], is then created and set to the original flux values. The liquid-steam boundary is located by finding instances when steam elements are adjacent to liquid elements. The arrangement of the element pairs is determined based on the location of the elements relative to the locations of the common nodes (See Table 2.1 in Section 2.3).

Using data available for the elements and nodes, the flux from the steam element and normal to the liquid-steam boundary is calculated according to the steps described in Section 2.3. The saturation temperature is interpolated from the temperature data for each of the two elements and then used to determine the enthalpy value using the enthalpy of vaporization calculated by the water property methods in Aquarius. The source/sink term is then calculated from the flux normal to the boundary, the enthalpy, and the fluid density for the steam element. The source/sink term is divided to the common nodes so that it can be added to the new flux array and included in global matrix. For validation and reference purposes, the "Steam Boundary" data table is printed to a file text file, SteamData, which is saved in the project files.

A4.2 Algorithm Validation

The following set of tables organize calculations of the normal component of the flux and the source/sink values and allows for all geometries and liquid/steam arrangements when the liquid-steam boundary is neither horizontal nor vertical. Lengths for each side of the element can be entered or calculated. The angle of rotation is calculated using Eq. 2.12. Eq. 2.13 is applied according to the slope of the liquid-steam

boundary. The normal component of the flux and the source/sink term are calculated as described above with values being multiplied by -1 when the liquid element is above the steam element.

Type 1: Negative Slope, Steam Above Liquid

	A	B	C	D
Lengths				
Horizontal				
Vertical				
Diagonal				
Angles				
tan(theta)				
theta				
Fluxes				
q(x)	+	-	+	-
q(y)	+	+	-	-
q				
"Normal" Flux				
q(n)				
Source/Sink Term				
Q(n)				

Type 2: Negative Slope, Liquid Above Steam

	A	B	C	D
Lengths				
Horizontal				
Vertical				
Diagonal				
Angles				
tan(theta)				
theta				
Fluxes				
q(x)	+	-	+	-
q(y)	+	+	-	-
q				
"Normal" Flux				
q(n)				
Source/Sink Term				
Q(n)	$-q(n)\Delta H$	$-q(n)\Delta H$	$-q(n)\Delta H$	$-q(n)\Delta H$

Type 3: Positive Slope, Steam Above Liquid

	A	B	C	D
Lengths				
Horizontal				
Vertical				
Diagonal				
Angles				
tan(theta)				
theta				
PI-theta				
Fluxes				
q(x)	+	-	+	-
q(y)	+	+	-	-
q				
"Normal" Flux				
q(n)				
Source/Sink Term				
Q(n)				

Type 4: Postive Slope, Liquid Above Steam				
	A	B	C	D
Lengths				
Horizontal				
Vertical				
Diagonal				
Angles				
tan(theta)				
theta				
PI-theta				
Fluxes				
q(x)	+	-	+	-
q(y)	+	+	-	-
q				
"Normal" Flux				
q(n)				
Source/Sink Term				
Q(n)	$-q(n)\Delta H$	$-q(n)\Delta H$	$-q(n)\Delta H$	$-q(n)\Delta H$

A4.3 C# Code

```
public void LatentHeat()

{
// Data Table to hold steam boundary information
DataTable steamBoundary = new DataTable("Steam Boundary");

#region //Data Columns
DataColumn node1 = new DataColumn(); //Of an elements nodes, whichever is first (0, 1, or 2)
node1.DataType = System.Type.GetType("System.Int32");
node1.ColumnName = "Node 1";
node1.ReadOnly = true;
steamBoundary.Columns.Add(node1);

DataColumn node2 = new DataColumn(); //Of an elements nodes, whichever is second (0, 1, or 2)
node2.DataType = System.Type.GetType("System.Int32");
node2.ColumnName = "Node 2";
node2.ReadOnly = true;
steamBoundary.Columns.Add(node2);

DataColumn steamElm = new DataColumn();
steamElm.DataType = System.Type.GetType("System.Int32");
steamElm.ColumnName = "Steam Element"; //Element "i"
steamElm.ReadOnly = true;
steamBoundary.Columns.Add(steamElm);
```

```
DataColumn liquidElm = new DataColumn();
liquidElm.DataType = System.Type.GetType("System.Int32");
liquidElm.ColumnName = "Liquid Element"; //Element "j"
liquidElm.ReadOnly = true;
steamBoundary.Columns.Add(liquidElm);

DataColumn steamTemp = new DataColumn();
steamTemp.DataType = System.Type.GetType("System.Decimal");
steamTemp.ColumnName = "Steam Temperature"; //Temperature of element "i"
steamTemp.ReadOnly = true;
steamBoundary.Columns.Add(steamTemp);

DataColumn liquidTemp = new DataColumn();
liquidTemp.DataType = System.Type.GetType("System.Decimal");
liquidTemp.ColumnName = "Liquid Temperature"; //Temperature of element "j"
liquidTemp.ReadOnly = true;
steamBoundary.Columns.Add(liquidTemp);

DataColumn transitionTemp = new DataColumn();
transitionTemp.DataType = System.Type.GetType("System.Decimal");
transitionTemp.ColumnName = "Transition Temperature"; //Saturation temperture between the
temperatures of nodes "i" and "j"
transitionTemp.ReadOnly = true;
steamBoundary.Columns.Add(transitionTemp);

DataColumn deltaHvap = new DataColumn();
deltaHvap.DataType = System.Type.GetType("System.Decimal");
deltaHvap.ColumnName = "Enthalpy Vaporization"; //"Delta H" determined from the look-up table using
the saturation or transition temperature
deltaHvap.ReadOnly = true;
steamBoundary.Columns.Add(deltaHvap);

DataColumn fluidFlux = new DataColumn();
fluidFlux.DataType = System.Type.GetType("System.Decimal");
fluidFlux.ColumnName = "Fluid Flux"; //Magnitude of the flux in node "i"
fluidFlux.ReadOnly = true;
steamBoundary.Columns.Add(fluidFlux);

DataColumn normalFlux = new DataColumn();
normalFlux.DataType = System.Type.GetType("System.Decimal");
normalFlux.ColumnName = "Normal Flux"; //Magnitude and direction of flux across the liquid-steam
interface
normalFlux.ReadOnly = true;
steamBoundary.Columns.Add(normalFlux);

DataColumn sourceSinkTerm = new DataColumn();
sourceSinkTerm.DataType = System.Type.GetType("System.Decimal");
sourceSinkTerm.ColumnName = "Source Sink Term"; //Magnitude and "direction" (i.e. source or sink) of
energy across the liquid-steam interface
sourceSinkTerm.ReadOnly = true;
steamBoundary.Columns.Add(sourceSinkTerm);

DataColumn nodeValue = new DataColumn();
nodeValue.DataType = System.Type.GetType("System.Decimal");
nodeValue.ColumnName = "Node Value"; //Magnitude and "direction" of energy divided to the common
nodes
```

```
nodeValue.ReadOnly = true;
steamBoundary.Columns.Add(nodeValue);

DataColumn traceCode = new DataColumn();
traceCode.DataType = System.Type.GetType("System.String");
traceCode.ColumnName = "Code"; //Code (letters A to U) to represent the end point within the decision
tree
traceCode.ReadOnly = true;
steamBoundary.Columns.Add(traceCode);

DataColumn elmDensity = new DataColumn();
elmDensity.DataType = System.Type.GetType("System.Decimal");
elmDensity.ColumnName = "Density"; //Density
elmDensity.ReadOnly = true;
steamBoundary.Columns.Add(elmDensity);

DataColumn fluxX = new DataColumn();
fluxX.DataType = System.Type.GetType("System.Decimal");
fluxX.ColumnName = "Flux X"; //X component of fluid flux
fluxX.ReadOnly = true;
steamBoundary.Columns.Add(fluxX);

DataColumn fluxY = new DataColumn();
fluxY.DataType = System.Type.GetType("System.Decimal");
fluxY.ColumnName = "Flux Y"; //Y component of fluid flux
fluxY.ReadOnly = true;
steamBoundary.Columns.Add(fluxY);

DataColumn theta = new DataColumn();
theta.DataType = System.Type.GetType("System.Decimal");
theta.ColumnName = "Theta"; //Angle of rotation
theta.ReadOnly = true;
steamBoundary.Columns.Add(theta);

DataColumn eLength = new DataColumn();
eLength.DataType = System.Type.GetType("System.Decimal");
eLength.ColumnName = "Length"; //Length of steam boundary
eLength.ReadOnly = true;
steamBoundary.Columns.Add(eLength);
#endregion

for (int k = 0; k < M.nNd; k++) //Set the new flux array equal to the original flux array
        lflx[k] = flx[k];

for (int i = 0; i < M.nElm; i++) // When steam boundary is within system boundaries
{
    if (water.pReg[i] == 3) // Steam elements
    {
        for (int j = 0; j < M.nElm; j++)
        {
        if ((water.pReg[j] == 1) || ((water.pReg[j] == 0) && (water.Tf[j] < 374))) //Liquid elements
            //The first (0) and second (1) nodes from the steam element are shared with two of the nodes
            from the liquid element
            if (((M.Nd[i, 0] == M.Nd[j, 0]) || (M.Nd[i, 0] == M.Nd[j, 1]) || (M.Nd[i, 0] == M.Nd[j, 2]))
            && ((M.Nd[i, 1] == M.Nd[j, 0]) || (M.Nd[i, 1] == M.Nd[j, 1]) || (M.Nd[i, 1] == M.Nd[j, 2])))
            {
```

```
DataRow boundaryRow = steamBoundary.NewRow(); //Populate the data table with
information about the liquid-steam interface
boundaryRow["Node 1"] = M.Nd[i, 0];
boundaryRow["Node 2"] = M.Nd[i, 1];
boundaryRow["Steam Element"] = i;
boundaryRow["Liquid Element"] = j;
boundaryRow["Steam Temperature"] = water.Tf[i];
boundaryRow["Liquid Temperature"] = water.Tf[j];
boundaryRow["Flux X"] = fluid.q[i, 0];
boundaryRow["Flux Y"] = fluid.q[i,1];
boundaryRow["Density"] = water.Dw[i];

double x, yz, m1, m2, b1, b2, disti, distj, rotation, elmLength;
double norm = 0; //Initialize the normal flux
 int direction; //+1 for flow from steam to liquid; -1 for flow from liquid to steam

#region // Vertical line between element centroids
if (M.eCd[i, 0] == M.eCd[j, 0]) //X-coordinates for the centroids of the steam and liquid
elements are the same
{
    #region // Horizontal line between common nodes
    if (M.nCd[M.Nd[i, 0], 1] == M.nCd[M.Nd[i, 1], 1]) //Y-coordinates of the two
    common nodes are the same
    {
        x = M.eCd[i, 0]; //Value for interpolation of saturation temperture below
        yz = M.nCd[M.Nd[i, 0], 1]; //Value for interpolation of saturation temperture
        below
        elmLength = Math.Abs(M.nCd[M.Nd[i, 0], 0] - M.nCd[M.Nd[i, 1], 0]); //The
        length of the liquid-steam interface

        #region //Steam above liquid
        if (M.eCd[i, 1] > M.eCd[j, 1]) //Y-coordinate for the centroid of the steam
        element is greater than the Y-coordinate of the liquid element
        {
            if (fluid.q[i, 1] < 0) //Y-component of fluid flux is negative
            {
                direction = 1; //Flow is from steam to liquid
                norm = Math.Abs(fluid.q[i, 1]) * direction * elmLength; //Flux across
                the boundary is the magnitude of the y-component of the fluid flux
                multiplied by the direction and the length of the liquid-steam interface
            }

            else if (fluid.q[i, 1] > 0) //Y-component of fluid flux is positive
            {
                direction = -1; //Flow is from liquid to steam
                 norm = Math.Abs(fluid.q[i, 1]) * direction * elmLength;
            }

            else if (fluid.q[i, 1] == 0) //Y-component of fluid flux is 0 (i.e. completely
            horizontal flow or no flow)
            {
                direction = 0; //No flow across the liquid-steam interface
                norm = Math.Abs(fluid.q[i, 1]) * direction * elmLength;
            }
        }
        #endregion
```

```
        #region //Liquid above steam
        else if (M.eCd[i, 1] < M.eCd[j, 1]) //Y-coordinate for the centroid of the steam
        element is less than the Y-coordinate of the liquid element
        {
            if (fluid.q[i, 1] < 0) //Y-component of fluid flux is negative
            {
                direction = -1; //Flow is from liquid to steam
                norm = Math.Abs(fluid.q[i, 1]) * direction * elmLength;
            }

            else if (fluid.q[i, 1] > 0) //Y-component of fluid flux is positive
            {
                direction = 1; //Flow is from steam to liquid
                norm = Math.Abs(fluid.q[i, 1]) * direction * elmLength;
            }

            else if (fluid.q[i, 1] == 0) //Y-component of fluid flux is 0 (i.e. completely
            horizontal flow or no flow)
            {
                direction = 0; //No flow across the liquid-steam interface
                norm = Math.Abs(fluid.q[i, 1]) * direction * elmLength;
            }
        }
        #endregion

    boundaryRow["Code"] = "A";//Endpoint for the decision tree
    }
    #endregion

    #region // Line between common nodes neither horizontal nor vertical
    else //Diagonal line between common nodes; line between nodes is not parallel or
    perpendicular to the coordinate system axes
    {
        x = M.eCd[i, 0]; //Value for interpolation of saturation temperture below
        m2 = (M.nCd[M.Nd[i, 1], 1] - M.nCd[M.Nd[i, 0], 1]) / (M.nCd[M.Nd[i, 1], 0] -
        M.nCd[M.Nd[i, 0], 0]); //Value for interpolation of saturation temperture below
        (slope of line)
        b2 = M.nCd[M.Nd[i, 0], 1] - (m2 * M.nCd[M.Nd[i, 0], 0]); //Value for
        interpolation of saturation temperture below (y intercept of line)
        yz = (m2 * x) + b2; //Value for interpolation of saturation temperture below
        elmLength = Math.Sqrt(((M.nCd[M.Nd[i, 0], 0] - M.nCd[M.Nd[i, 1], 0]) *
        (M.nCd[M.Nd[i, 0], 0] - M.nCd[M.Nd[i, 1], 0])) + ((M.nCd[M.Nd[i, 0], 1] -
        M.nCd[M.Nd[i, 1], 1]) * (M.nCd[M.Nd[i, 0], 1] - M.nCd[M.Nd[i, 1], 1])));
        //Length of liquid-steam interface
        boundaryRow["Length"] = elmLength;

        rotation = (Math.Atan((Math.Abs(M.nCd[M.Nd[i, 0], 0] - M.nCd[M.Nd[i, 1],
        0])) / (Math.Abs(M.nCd[M.Nd[i, 0], 1] - M.nCd[M.Nd[i, 1], 1]))));
        //(Above) Magnitude of the angle between the line between the common nodes
        (the "x-axis" for the local coordinate system) and the x-axis of the global
        coordinate system
        boundaryRow["Theta"] = rotation;

        #region //Negative slope for boundary
```

```
if (M.nCd[M.Nd[i, 0], 1] > M.nCd[M.Nd[i, 1], 1]) //Y coordinate of first
common node greater than the y-coordinate of the second common node
{

    #region //Steam above liquid
    //Y-coordinate of steam element in the local coordinate system is greater
    than the y-coordinate of teh liquid element in the local coordinate system
     if (M.eCd[i, 1] > M.eCd[j, 1])
     {
         direction = -1;
         norm = (fluid.q[i, 0] * Math.Cos(rotation) + fluid.q[i, 1] *
         Math.Sin(rotation))*direction*elmLength;
         boundaryRow["Code"] = "B-S";
     }
     #endregion

     #region //Liquid above steam
     else if (M.eCd[i, 1] < M.eCd[j, 1])
     {
         direction = 1;
         norm = (fluid.q[i, 0] * Math.Cos(rotation) + fluid.q[i, 1] *
         Math.Sin(rotation)) * direction * elmLength;
         boundaryRow["Code"] = "B-L";
     }
     #endregion
}
#endregion

#region //Postive slope for boundary
else if (M.nCd[M.Nd[i, 0], 1] < M.nCd[M.Nd[i, 1], 1]) //Y coordinate of first
common node less than the y-coordinate of the second common node
{

    #region //Steam above liquid
    if (M.eCd[i, 1] > M.eCd[j, 1])
    {
        direction = -1;
        norm = (fluid.q[i, 0] * Math.Cos(Math.PI-rotation) + fluid.q[i, 1] *
        Math.Sin(Math.PI-rotation)) * direction*elmLength;
        boundaryRow["Code"] = "B+S";
    }
    #endregion

    #region //Liquid above steam
    else if (M.eCd[i,1] < M.eCd[j,1])
    {
        direction = 1;
        norm = (fluid.q[i, 0] * Math.Cos(Math.PI - rotation) + fluid.q[i, 1] *
        Math.Sin(Math.PI - rotation)) * direction * elmLength;
        boundaryRow["Code"] = "B+L";
    }
    #endregion
}
#endregion
}
#endregion
```

```
        //Flux values -- added to data table
        boundaryRow["Fluid Flux"] = Math.Sqrt((fluid.q[i, 0] * fluid.q[i, 0]) + (fluid.q[i, 1]
        * fluid.q[i, 1]));
        boundaryRow["Normal Flux"] = norm;

        // Interpolate for saturation temperature -- add values to data table
        disti = Math.Sqrt(((yz - M.eCd[i, 1]) * (yz - M.eCd[i, 1])) + ((x - M.eCd[i, 0]) * (x -
        M.eCd[i, 0])));
        distj = Math.Sqrt(((yz - M.eCd[j, 1]) * (yz - M.eCd[j, 1])) + ((x - M.eCd[j, 0]) * (x -
        M.eCd[j, 0])));

        double satTemp = water.Tf[i] + ((0 - (-disti)) / (distj - (-disti))) * (water.Tf[j] -
        water.Tf[i]);

        boundaryRow["Transition Temperature"] = satTemp;
        boundaryRow["Enthalpy Vaporization"] =
        water.EnthalpyOfVapourization(satTemp);

        //Calculate Source/Sink Term -- added to data table
        double sourceSink = norm * water.EnthalpyOfVapourization(satTemp) *
        water.Dw[i];
        boundaryRow["Source Sink Term"] = sourceSink;

        //Divide to Nodes
        if (M.CCode == 0)
        {
            lflx[M.Nd[i, 0]] = lflx[M.Nd[i, 0]] + (sourceSink / 2);
            lflx[M.Nd[i, 1]] = lflx[M.Nd[i, 1]] + (sourceSink / 2);
            boundaryRow["Node Value"] = sourceSink/2;
        }

        else if (M.CCode == 1)
        {
            lflx[M.Nd[i, 0]] = lflx[M.Nd[i, 0]] + (sourceSink / 2) * 2 * Math.PI *
            M.nCd[M.Nd[i, 0], 0];
            lflx[M.Nd[i, 1]] = lflx[M.Nd[i, 1]] + (sourceSink / 2) * 2 * Math.PI *
            M.nCd[M.Nd[i, 1], 0];
            boundaryRow["Node Value"] = sourceSink / 2 * 2 * Math.PI * M.nCd[M.Nd[i,
            0], 0];
        }
    }
    #endregion

    #region // Horizontal line between element centroids
    else if (M.eCd[i, 1] == M.eCd[j, 1]) //Y-coordinates of steam and liquid elements are the
    same
    {
        #region // Vertical line between common nodes
        if (M.nCd[M.Nd[i, 0], 0] == M.nCd[M.Nd[i, 1], 0]) //X-coordinates of the common
        nodes are the same
        {
            x = M.nCd[M.Nd[i, 0], 0];
            yz = M.eCd[i, 1];
            elmLength = Math.Abs(M.nCd[M.Nd[i, 0], 1] - M.nCd[M.Nd[i, 1], 1]);
```

```
        #region //Steam right of liquid
        if (M.eCd[i,0] > M.eCd[j,0])
        {
            if (fluid.q[i, 0] < 0)
            {
                direction = 1;
                norm = Math.Abs(fluid.q[i, 0]) * direction * elmLength;
            }

            else if (fluid.q[i,0] > 0)
            {
                direction = -1;
                norm = Math.Abs(fluid.q[i, 0]) * direction * elmLength;
            }

            else if (fluid.q[i, 0] == 0)
            {
                direction = 0;
                norm = Math.Abs(fluid.q[i, 0]) * direction * elmLength;
            }
        }
        #endregion

        #region //Liquid right of steam
        else if (M.eCd[i,0] < M.eCd[j,0])
        {
            if (fluid.q[i, 0] < 0)
            {
                direction = -1;
                norm = Math.Abs(fluid.q[i, 0]) * direction * elmLength;
            }

            else if (fluid.q[i, 0] > 0)
            {
                direction = 1;
                norm = Math.Abs(fluid.q[i, 0]) * direction * elmLength;
            }

            else if (fluid.q[i, 0] == 0)
            {
                direction = 0;
                norm = Math.Abs(fluid.q[i, 0]) * direction * elmLength;
            }
        }
        #endregion

        boundaryRow["Code"] = "C";
    }
    #endregion

    #region // Line between common nodes neither horizontal nor vertical
    else
    {
        yz = M.eCd[i, 1];
        m2 = (M.nCd[M.Nd[i, 1], 1] - M.nCd[M.Nd[i, 0], 1]) / (M.nCd[M.Nd[i, 1], 0] -
        M.nCd[M.Nd[i, 0], 0]);
```

```
b2 = M.nCd[M.Nd[i, 0], 1] - (m2 * M.nCd[M.Nd[i, 0], 0]);
x = (yz - b2) / m2;
elmLength = Math.Sqrt(((M.nCd[M.Nd[i, 0], 0] - M.nCd[M.Nd[i, 1], 0]) *
(M.nCd[M.Nd[i, 0], 0] - M.nCd[M.Nd[i, 1], 0])) + ((M.nCd[M.Nd[i, 0], 1] -
M.nCd[M.Nd[i, 1], 1]) * (M.nCd[M.Nd[i, 0], 1] - M.nCd[M.Nd[i, 1], 1])));
boundaryRow["Length"] = elmLength;

rotation = (Math.Atan((Math.Abs(M.nCd[M.Nd[i, 0], 0] - M.nCd[M.Nd[i, 1],
0])) / Math.Abs((M.nCd[M.Nd[i, 0], 1] - M.nCd[M.Nd[i, 1], 1]))));
boundaryRow["Theta"] = rotation;

#region //Negative slope for boundary
if (M.nCd[M.Nd[i, 0], 1] > M.nCd[M.Nd[i, 1], 1])
{
    #region //Steam above liquid
    if (M.eCd[i, 0] > M.eCd[j, 0])
    {
        direction = -1;
        norm = (fluid.q[i, 0] * Math.Cos(rotation) + fluid.q[i, 1] *
        Math.Sin(rotation)) * direction * elmLength;
        boundaryRow["Code"] = "D-S";
    }
    #endregion

    #region //Liquid above steam
    else if (M.eCd[i, 0] < M.eCd[j, 0])
    {
        direction = 1;
        norm = (fluid.q[i, 0] * Math.Cos(rotation) + fluid.q[i, 1] *
        Math.Sin(rotation)) * direction * elmLength;
        boundaryRow["Code"] = "D-L";
    }
    #endregion
}
#endregion

#region //Postive slope for boundary
else if (M.nCd[M.Nd[i, 0], 1] < M.nCd[M.Nd[i, 1], 1])
{
    #region //Steam above liquid
    if (M.eCd[i, 0] < M.eCd[j, 0])
    {
        direction = -1;
        norm = (fluid.q[i, 0] * Math.Cos(Math.PI - rotation) + fluid.q[i, 1] *
        Math.Sin(Math.PI - rotation)) * direction * elmLength;
        boundaryRow["Code"] = "D+S";
    }
    #endregion

    #region //Liquid above steam
    else if (M.eCd[i, 0] > M.eCd[j, 0])
    {
        direction = 1;
        norm = (fluid.q[i, 0] * Math.Cos(Math.PI - rotation) + fluid.q[i, 1] *
        Math.Sin(Math.PI - rotation)) * direction * elmLength;
        boundaryRow["Code"] = "D+L";
```

```
                }
                #endregion
            }
            #endregion
        }
        #endregion

        //Flux values
        boundaryRow["Fluid Flux"] = Math.Sqrt((fluid.q[i, 0] * fluid.q[i, 0]) + (fluid.q[i, 1]
        * fluid.q[i, 1]));
        boundaryRow["Normal Flux"] = norm;

        //Interpolate for saturation temperature
        disti = Math.Sqrt(((yz - M.eCd[i, 1]) * (yz - M.eCd[i, 1])) + ((x - M.eCd[i, 0]) * (x -
        M.eCd[i, 0])));
        distj = Math.Sqrt(((yz - M.eCd[j, 1]) * (yz - M.eCd[j, 1])) + ((x - M.eCd[j, 0]) * (x -
        M.eCd[j, 0])));

        double satTemp = water.Tf[i] + ((0 - (-disti)) / (distj - (-disti))) * (water.Tf[j] -
        water.Tf[i]);

        boundaryRow["Transition Temperature"] = satTemp;
        boundaryRow["Enthalpy Vaporization"] =
        water.EnthalpyOfVapourization(satTemp);

        //Calculate Source/Sink Term
        double sourceSink = norm * water.EnthalpyOfVapourization(satTemp) *
        water.Dw[i];
        boundaryRow["Source Sink Term"] = sourceSink;

        //Divide to Nodes
        if (M.CCode == 0)
        {
            lflx[M.Nd[i, 0]] = lflx[M.Nd[i, 0]] + (sourceSink / 2);
            lflx[M.Nd[i, 1]] = lflx[M.Nd[i, 1]] + (sourceSink / 2);
            boundaryRow["Node Value"] = sourceSink / 2;
        }

        else if (M.CCode == 1)
        {
            lflx[M.Nd[i, 0]] = lflx[M.Nd[i, 0]] + (sourceSink / 2) * 2 * Math.PI *
            M.nCd[M.Nd[i, 0], 0];
            lflx[M.Nd[i, 1]] = lflx[M.Nd[i, 1]] + (sourceSink / 2) * 2 * Math.PI *
            M.nCd[M.Nd[i, 1], 0];
            boundaryRow["Node Value"] = sourceSink / 2 * 2 * Math.PI * M.nCd[M.Nd[i,
            0], 0];
        }
    }
    #endregion

    #region // Line between element centroids neither horizontal nor vertical
    else //Diagonal line between element centroids
    {
        #region // Horizontal line between common nodes
        if (M.nCd[M.Nd[i, 0], 1] == M.nCd[M.Nd[i, 1], 1])
        {
```

```
        yz = M.nCd[M.Nd[i, 0], 1];
        m1 = (M.eCd[j, 1] - M.eCd[i, 1]) / (M.eCd[j, 0] - M.eCd[i, 0]);
        b1 = M.eCd[i, 1] - (m1 * M.eCd[i, 0]);
        x = (yz - b1) / m1;
        elmLength = Math.Abs(M.nCd[M.Nd[i, 0], 0] - M.nCd[M.Nd[i, 1], 0]);

        #region //Steam above liquid
        if (M.eCd[i, 1] > M.eCd[j, 1])
        {
            if (fluid.q[i, 1] < 0)
            {
                direction = 1;
                norm = Math.Abs(fluid.q[i, 1]) * direction * elmLength;
            }

            else if (fluid.q[i, 1] > 0)
            {
                direction = -1;
                norm = Math.Abs(fluid.q[i, 1]) * direction * elmLength;
             }

            else if (fluid.q[i, 1] == 0)
            {
                direction = 0;
                norm = Math.Abs(fluid.q[i, 1]) * direction * elmLength;
            }
        }
        #endregion

        #region //Liquid above steam
        else if (M.eCd[i, 1] < M.eCd[j, 1])
        {
            if (fluid.q[i, 1] < 0)
            {
                direction = -1;
                norm = Math.Abs(fluid.q[i, 1]) * direction * elmLength;
             }

            else if (fluid.q[i, 1] > 0)
            {
                direction = 1;
                 norm = Math.Abs(fluid.q[i, 1]) * direction * elmLength;
             }

            else if (fluid.q[i, 1] == 0)
            {
                direction = 0;
                norm = Math.Abs(fluid.q[i, 1]) * direction * elmLength;
             }
        }
        #endregion

        boundaryRow["Code"] = "E";
    }
    #endregion
```

```
#region // Vertical line between common nodes
else if (M.nCd[M.Nd[i, 0], 0] == M.nCd[M.Nd[i, 1], 0])
{
    x = M.nCd[M.Nd[i, 0], 0];
    m1 = (M.eCd[j, 1] - M.eCd[i, 1]) / (M.eCd[j, 0] - M.eCd[i, 0]);
    b1 = M.eCd[i, 1] - (m1 * M.eCd[i, 0]);
    yz = (m1 * x) + b1;
    elmLength = Math.Abs(M.nCd[M.Nd[i, 0], 1] - M.nCd[M.Nd[i, 1], 1]);

    #region //Steam to right of liquid
    if (M.eCd[i, 0] > M.eCd[j, 0])
    {
        if (fluid.q[i, 0] < 0)
        {
            direction = 1;
            norm = Math.Abs(fluid.q[i, 0]) * direction * elmLength;
        }

        else if (fluid.q[i, 0] > 0)
        {
            direction = -1;
            norm = Math.Abs(fluid.q[i, 0]) * direction * elmLength;
        }

        else if (fluid.q[i, 0] == 0)
        {
            direction = 0;
            norm = Math.Abs(fluid.q[i, 0]) * direction * elmLength;
        }
    }
    #endregion

    #region //Liquid to right of steam
    else if (M.eCd[i, 0] < M.eCd[j, 0])
    {
        if (fluid.q[i, 0] < 0)
        {
            direction = -1;
            norm = Math.Abs(fluid.q[i, 0]) * direction * elmLength;
        }

        else if (fluid.q[i, 0] > 0)
        {
            direction = 1;
            norm = Math.Abs(fluid.q[i, 0]) * direction * elmLength;
        }

        else if (fluid.q[i, 0] == 0)
        {
            direction = 0;
            norm = Math.Abs(fluid.q[i, 0]) * direction * elmLength;
        }
    }
    #endregion

    boundaryRow["Code"] = "F";
```

```
}
#endregion

#region // Line between common nodes neither horizontal nor vertical
else
{
    m1 = (M.eCd[j, 1] - M.eCd[i, 1]) / (M.eCd[j, 0] - M.eCd[i, 0]);
    m2 = (M.nCd[M.Nd[i, 1], 1] - M.nCd[M.Nd[i, 0], 1]) / (M.nCd[M.Nd[i, 1], 0] -
    M.nCd[M.Nd[i, 0], 0]);
    b1 = M.eCd[i, 1] - (m1 * M.eCd[i, 0]);
    b2 = M.nCd[M.Nd[i, 0], 1] - (m2 * M.nCd[M.Nd[i, 0], 0]);
    x = (b2 - b1) / (m1 - m2);
    yz = (m1 * x) + b1;
    elmLength = Math.Sqrt(((M.nCd[M.Nd[i, 0], 0] - M.nCd[M.Nd[i, 1], 0]) *
    (M.nCd[M.Nd[i, 0], 0] - M.nCd[M.Nd[i, 1], 0])) + ((M.nCd[M.Nd[i, 0], 1] -
    M.nCd[M.Nd[i, 1], 1]) * (M.nCd[M.Nd[i, 0], 1] - M.nCd[M.Nd[i, 1], 1])));
    boundaryRow["Length"] = elmLength;

    rotation = (Math.Atan((Math.Abs(M.nCd[M.Nd[i, 0], 0] - M.nCd[M.Nd[i, 1],
    0])) / Math.Abs((M.nCd[M.Nd[i, 0], 1] - M.nCd[M.Nd[i, 1], 1]))));
    boundaryRow["Theta"] = rotation;

    #region // Negative slope for boundary
    if (M.nCd[M.Nd[i, 0], 1] > M.nCd[M.Nd[i, 1], 1])
    {
        #region //Steam above liquid
        if (M.eCd[i, 0] > M.eCd[j, 0])
        {
            direction = -1;
            norm = (fluid.q[i, 0] * Math.Cos(rotation) + fluid.q[i, 1] *
            Math.Sin(rotation)) * direction * elmLength;
            boundaryRow["Code"] = "G-S";
        }
        #endregion

        #region //Liquid above steam
        else if (M.eCd[i, 0] < M.eCd[j, 0])
        {
            direction = 1;
            norm = (fluid.q[i, 0] * Math.Cos(rotation) + fluid.q[i, 1] *
            Math.Sin(rotation)) * direction * elmLength;
            boundaryRow["Code"] = "G-L";
        }
        #endregion
    }
    #endregion

    #region //Postive slope for boundary
    else if (M.nCd[M.Nd[i, 0], 1] < M.nCd[M.Nd[i, 1], 1])
    {
        #region //Steam above liquid
        if (M.eCd[i, 0] < M.eCd[j, 0])
        {
            direction = -1;
            norm = (fluid.q[i, 0] * Math.Cos(Math.PI - rotation) + fluid.q[i, 1] *
            Math.Sin(Math.PI - rotation)) * direction * elmLength;
```

```
                boundaryRow["Code"] = "G+S";
            }
            #endregion

            #region //Liquid above steam
            else if (M.eCd[i, 0] > M.eCd[j, 0])
            {
                direction = 1;
                norm = (fluid.q[i, 0] * Math.Cos(Math.PI - rotation) + fluid.q[i, 1] *
                Math.Sin(Math.PI - rotation)) * direction * elmLength;
                boundaryRow["Code"] = "G+L";
            }
            #endregion
        }
        #endregion
    }
    #endregion

    //Flux values
    boundaryRow["Fluid Flux"] = Math.Sqrt((fluid.q[i, 0] * fluid.q[i, 0]) + (fluid.q[i, 1]
    * fluid.q[i, 1]));
    boundaryRow["Normal Flux"] = norm;

    // Interpolate for saturation temperature
    disti = Math.Sqrt(((yz - M.eCd[i, 1]) * (yz - M.eCd[i, 1])) + ((x - M.eCd[i, 0]) * (x -
    M.eCd[i, 0])));
    distj = Math.Sqrt(((yz - M.eCd[j, 1]) * (yz - M.eCd[j, 1])) + ((x - M.eCd[j, 0]) * (x -
    M.eCd[j, 0])));

    double satTemp = water.Tf[i] + ((0 - (-disti)) / (distj - (-disti))) * (water.Tf[j] -
    water.Tf[i]);

    boundaryRow["Transition Temperature"] = satTemp;
    boundaryRow["Enthalpy Vaporization"] =
    water.EnthalpyOfVapourization(satTemp);

    //Calculate Source/Sink Term
    double sourceSink = norm * water.EnthalpyOfVapourization(satTemp) *
    water.Dw[i];
    boundaryRow["Source Sink Term"] = sourceSink;

    //Divide to Nodes
    if (M.CCode == 0)
    {
        lflx[M.Nd[i, 0]] = lflx[M.Nd[i, 0]] + (sourceSink / 2);
        lflx[M.Nd[i, 1]] = lflx[M.Nd[i, 1]] + (sourceSink / 2);
        boundaryRow["Node Value"] = sourceSink / 2;
    }

    else if (M.CCode == 1)
    {
        lflx[M.Nd[i, 0]] = lflx[M.Nd[i, 0]] + (sourceSink / 2) * 2 * Math.PI *
        M.nCd[M.Nd[i, 0], 0];
        lflx[M.Nd[i, 1]] = lflx[M.Nd[i, 1]] + (sourceSink / 2) * 2 * Math.PI *
        M.nCd[M.Nd[i, 1], 0];
```

```
                boundaryRow["Node Value"] = sourceSink / 2 * 2 * Math.PI * M.nCd[M.Nd[i,
                0], 0];
            }
        }
        #endregion

        steamBoundary.Rows.Add(boundaryRow);
}

//The first (0) and third (2) nodes from the steam element are shared with two of the nodes
from the liquid element
else if (((M.Nd[i, 0] == M.Nd[j, 0]) || (M.Nd[i, 0] == M.Nd[j, 1]) || (M.Nd[i, 0] == M.Nd[j,
2])) && ((M.Nd[i, 2] == M.Nd[j, 0]) || (M.Nd[i, 2] == M.Nd[j, 1]) || (M.Nd[i, 2] == M.Nd[j,
2])))
{
    DataRow boundaryRow = steamBoundary.NewRow();
    boundaryRow["Node 1"] = M.Nd[i, 0];
    boundaryRow["Node 2"] = M.Nd[i, 2];
    boundaryRow["Steam Element"] = i;
    boundaryRow["Liquid Element"] = j;
    boundaryRow["Steam Temperature"] = water.Tf[i];
    boundaryRow["Liquid Temperature"] = water.Tf[j];
    boundaryRow["Flux X"] = fluid.q[i, 0];
    boundaryRow["Flux Y"] = fluid.q[i, 1];
    boundaryRow["Density"] = water.Dw[i];

    double x, yz, m1, m2, b1, b2, disti, distj, rotation, elmLength;
    double norm = 0;
    int direction;

    #region // Vertical line between element centroids
    if (M.eCd[i, 0] == M.eCd[j, 0]) // Vertical line between element centroids
    {
        #region //Horizontal line between common nodes
        if (M.nCd[M.Nd[i, 0], 1] == M.nCd[M.Nd[i, 2], 1])
        {
            x = M.eCd[i, 0];
            yz = M.nCd[M.Nd[i, 0], 1];
            elmLength = Math.Abs(M.nCd[M.Nd[i, 0], 0] - M.nCd[M.Nd[i, 2], 0]);

            #region //Steam above liquid
            if (M.eCd[i, 1] > M.eCd[j, 1])
            {
                if (fluid.q[i, 1] < 0)
                {
                    direction = 1;
                    norm = Math.Abs(fluid.q[i, 1]) * direction * elmLength;
                }

                else if (fluid.q[i, 1] > 0)
                {
                    direction = -1;
                    norm = Math.Abs(fluid.q[i, 1]) * direction * elmLength;
                }

                else if (fluid.q[i, 1] == 0)
```

```
            {
                direction = 0;
                norm = Math.Abs(fluid.q[i, 1]) * direction * elmLength;
            }
        }
        #endregion

        #region //Liquid above steam
        else if (M.eCd[i, 1] < M.eCd[j, 1])
        {
            if (fluid.q[i, 1] < 0)
            {
                direction = -1;
                norm = Math.Abs(fluid.q[i, 1]) * direction * elmLength;
            }

            else if (fluid.q[i, 1] > 0)
            {
                direction = 1;
                norm = Math.Abs(fluid.q[i, 1]) * direction * elmLength;
            }

            else if (fluid.q[i, 1] == 0)
            {
                direction = 0;
                norm = Math.Abs(fluid.q[i, 1]) * direction * elmLength;
            }
        }
        #endregion

        boundaryRow["Code"] = "H";
    }
    #endregion

    #region // Line between common nodes neither horizontal nor vertical
    else
    {
        x = M.eCd[i, 0];
        m2 = (M.nCd[M.Nd[i, 2], 1] - M.nCd[M.Nd[i, 0], 1]) / (M.nCd[M.Nd[i, 2], 0] -
        M.nCd[M.Nd[i, 0], 0]);
        b2 = M.nCd[M.Nd[i, 0], 1] - (m2 * M.nCd[M.Nd[i, 0], 0]);
        yz = (m2 * x) + b2;
        elmLength = Math.Sqrt(((M.nCd[M.Nd[i, 0], 0] - M.nCd[M.Nd[i, 2], 0]) *
        (M.nCd[M.Nd[i, 0], 0] - M.nCd[M.Nd[i, 2], 0])) + ((M.nCd[M.Nd[i, 0], 1] -
        M.nCd[M.Nd[i, 2], 1]) * (M.nCd[M.Nd[i, 0], 1] - M.nCd[M.Nd[i, 2], 1])));
        boundaryRow["Length"] = elmLength;

        rotation = (Math.Atan((Math.Abs(M.nCd[M.Nd[i, 0], 0] - M.nCd[M.Nd[i, 2],
        0])) / Math.Abs((M.nCd[M.Nd[i, 0], 1] - M.nCd[M.Nd[i, 2], 1]))));
        boundaryRow["Theta"] = rotation;

        #region //Negative slope for boundary
        if (M.nCd[M.Nd[i, 0], 1] > M.nCd[M.Nd[i, 2], 1])
        {
            #region //Steam above liquid
            if (M.eCd[i, 1] > M.eCd[j, 1])
```

```
            {
                direction = -1;
                norm = (fluid.q[i, 0] * Math.Cos(rotation) + fluid.q[i, 1] *
                Math.Sin(rotation)) * direction * elmLength;
                boundaryRow["Code"] = "I-S";
            }
            #endregion

            #region //Liquid above steam
            else if (M.eCd[i, 1] < M.eCd[j, 1])
            {
                direction = 1;
                norm = (fluid.q[i, 0] * Math.Cos(rotation) + fluid.q[i, 1] *
                Math.Sin(rotation)) * direction * elmLength;
                boundaryRow["Code"] = "I-L";
            }
            #endregion
        }
        #endregion

        #region //Positive slope for boundary
        else if (M.nCd[M.Nd[i, 0], 1] < M.nCd[M.Nd[i, 2], 1])
        {
            #region //Steam above liquid
            if (M.eCd[i, 1] > M.eCd[j, 1])
            {
                direction = -1;
                norm = (fluid.q[i, 0] * Math.Cos(Math.PI - rotation) + fluid.q[i, 1] *
                Math.Sin(Math.PI - rotation)) * direction * elmLength;
                boundaryRow["Code"] = "I+S";
            }
            #endregion

            #region //Liquid above steam
            else if (M.eCd[i, 1] < M.eCd[j, 1])
            {
                direction = 1;
                norm = (fluid.q[i, 0] * Math.Cos(Math.PI - rotation) + fluid.q[i, 1] *
                Math.Sin(Math.PI - rotation)) * direction * elmLength;
                boundaryRow["Code"] = "I+L";
            }
            #endregion
        }
        #endregion
    }
    #endregion

    //Flux values
    boundaryRow["Fluid Flux"] = Math.Sqrt((fluid.q[i, 0] * fluid.q[i, 0]) + (fluid.q[i, 1]
    * fluid.q[i, 1]));
    boundaryRow["Normal Flux"] = norm;

    // Interpolate for saturation temperature
    disti = Math.Sqrt(((yz - M.eCd[i, 1]) * (yz - M.eCd[i, 1])) + ((x - M.eCd[i, 0]) * (x -
    M.eCd[i, 0])));
```

```
        distj = Math.Sqrt(((yz - M.eCd[j, 1]) * (yz - M.eCd[j, 1])) + ((x - M.eCd[j, 0]) * (x -
        M.eCd[j, 0])));

        double satTemp = water.Tf[i] + ((0 - (-disti)) / (distj - (-disti))) * (water.Tf[j] -
        water.Tf[i]);

        boundaryRow["Transition Temperature"] = satTemp;
        boundaryRow["Enthalpy Vaporization"] =
        water.EnthalpyOfVapourization(satTemp);

        //Calculate Source/Sink Term
        double sourceSink = norm * water.EnthalpyOfVapourization(satTemp) *
        water.Dw[i];
        boundaryRow["Source Sink Term"] = sourceSink;

        //Divide to Nodes
        if (M.CCode == 0)
        {
            lflx[M.Nd[i, 0]] = lflx[M.Nd[i, 0]] + (sourceSink / 2);
            lflx[M.Nd[i, 2]] = lflx[M.Nd[i, 2]] + (sourceSink / 2);
            boundaryRow["Node Value"] = sourceSink / 2;
        }

        else if (M.CCode == 1)
        {
            lflx[M.Nd[i, 0]] = lflx[M.Nd[i, 0]] + (sourceSink / 2) * 2 * Math.PI *
            M.nCd[M.Nd[i, 0], 0];
            lflx[M.Nd[i, 2]] = lflx[M.Nd[i, 2]] + (sourceSink / 2) * 2 * Math.PI *
            M.nCd[M.Nd[i, 2], 0];
            boundaryRow["Node Value"] = sourceSink / 2 * 2 * Math.PI * M.nCd[M.Nd[i,
            0], 0];
        }
    }
    #endregion

    #region // Horizontal line between element centroids
    else if (M.eCd[i, 1] == M.eCd[j, 1])
    {
        #region // Vertical line between common nodes
        if (M.nCd[M.Nd[i, 0], 0] == M.nCd[M.Nd[i, 2], 0])
        {
            x = M.nCd[M.Nd[i, 0], 0];
            yz = M.eCd[i, 1];
            elmLength = Math.Abs(M.nCd[M.Nd[i, 0], 1] - M.nCd[M.Nd[i, 2], 1]);

            #region //Steam right of liquid
            if (M.eCd[i, 0] > M.eCd[j, 0])
            {
                if (fluid.q[i, 0] < 0)
                {
                    direction = 1;
                    norm = Math.Abs(fluid.q[i, 0]) * direction * elmLength;
                }

                else if (fluid.q[i, 0] > 0)
                {
```

```
                direction = -1;
                norm = Math.Abs(fluid.q[i, 0]) * direction * elmLength;
            }

            else if (fluid.q[i, 0] == 0)
            {
                direction = 0;
                norm = Math.Abs(fluid.q[i, 0]) * direction * elmLength;
            }
        }
        #endregion

        #region //Liquid right of steam
        else if (M.eCd[i, 0] < M.eCd[j, 0])
        {
            if (fluid.q[i, 0] < 0)
            {
                direction = -1;
                norm = Math.Abs(fluid.q[i, 0]) * direction * elmLength;
            }

            else if (fluid.q[i, 0] > 0)
            {
                direction = 1;
                norm = Math.Abs(fluid.q[i, 0]) * direction * elmLength;
            }

            else if (fluid.q[i, 0] == 0)
            {
                direction = 0;
                norm = Math.Abs(fluid.q[i, 0]) * direction * elmLength;
            }
        }
        #endregion

        boundaryRow["Code"] = "J";
    }
    #endregion

    #region // Line between common nodes neither horizontal nor vertical
    else
    {
        yz = M.eCd[i, 1];
        m2 = (M.nCd[M.Nd[i, 2], 1] - M.nCd[M.Nd[i, 0], 1]) / (M.nCd[M.Nd[i, 2], 0] -
        M.nCd[M.Nd[i, 0], 0]);
        b2 = M.nCd[M.Nd[i, 0], 1] - (m2 * M.nCd[M.Nd[i, 0], 0]);
        x = (yz - b2) / m2;
        elmLength = Math.Sqrt(((M.nCd[M.Nd[i, 0], 0] - M.nCd[M.Nd[i, 2], 0]) *
        (M.nCd[M.Nd[i, 0], 0] - M.nCd[M.Nd[i, 2], 0])) + ((M.nCd[M.Nd[i, 0], 1] -
        M.nCd[M.Nd[i, 2], 1]) * (M.nCd[M.Nd[i, 0], 1] - M.nCd[M.Nd[i, 2], 1])));
        boundaryRow["Length"] = elmLength;

        rotation = (Math.Atan((Math.Abs(M.nCd[M.Nd[i, 0], 0] - M.nCd[M.Nd[i, 2],
        0])) / Math.Abs((M.nCd[M.Nd[i, 0], 1] - M.nCd[M.Nd[i, 2], 1]))));
        boundaryRow["Theta"] = rotation;
```

```
        #region //Negative slope for boundary
        if (M.nCd[M.Nd[i, 0], 1] > M.nCd[M.Nd[i, 2], 1])
        {
            #region //Steam above liquid
            if (M.eCd[i, 0] > M.eCd[j, 0])
            {
                direction = -1;
                norm = (fluid.q[i, 0] * Math.Cos(rotation) + fluid.q[i, 1] *
                Math.Sin(rotation)) * direction * elmLength;
                boundaryRow["Code"] = "K-S";
            }
            #endregion

            #region //Liquid above steam
            else if (M.eCd[i, 0] < M.eCd[j, 0])
            {
                direction = 1;
                norm = (fluid.q[i, 0] * Math.Cos(rotation) + fluid.q[i, 1] *
                Math.Sin(rotation)) * direction * elmLength;
                boundaryRow["Code"] = "K-L";
            }
            #endregion
        }
        #endregion

        #region //Postive slope for boundary
        else if (M.nCd[M.Nd[i, 0], 1] < M.nCd[M.Nd[i, 2], 1])
        {
            #region //Steam above liquid
            if (M.eCd[i, 0] < M.eCd[j, 0])
            {
                direction = -1;
                norm = (fluid.q[i, 0] * Math.Cos(Math.PI - rotation) + fluid.q[i, 1] *
                Math.Sin(Math.PI - rotation)) * direction * elmLength;
                boundaryRow["Code"] = "K+S";
            }
            #endregion

            #region //Liquid above steam
            else if (M.eCd[i, 0] > M.eCd[j, 0])
            {
                direction = 1;
                norm = (fluid.q[i, 0] * Math.Cos(Math.PI - rotation) + fluid.q[i, 1] *
                Math.Sin(Math.PI - rotation)) * direction * elmLength;
                boundaryRow["Code"] = "K+L";
            }
            #endregion
        }
        #endregion
    }
    #endregion

    //Flux values
    boundaryRow["Fluid Flux"] = Math.Sqrt((fluid.q[i, 0] * fluid.q[i, 0]) + (fluid.q[i, 1]
    * fluid.q[i, 1]));
    boundaryRow["Normal Flux"] = norm;
```

```
        // Interpolate for saturation temperature
        disti = Math.Sqrt(((yz - M.eCd[i, 1]) * (yz - M.eCd[i, 1])) + ((x - M.eCd[i, 0]) * (x -
        M.eCd[i, 0])));
        distj = Math.Sqrt(((yz - M.eCd[j, 1]) * (yz - M.eCd[j, 1])) + ((x - M.eCd[j, 0]) * (x -
        M.eCd[j, 0])));

        double satTemp = water.Tf[i] + ((0 - (-disti)) / (distj - (-disti))) * (water.Tf[j] -
        water.Tf[i]);

        boundaryRow["Transition Temperature"] = satTemp;
        boundaryRow["Enthalpy Vaporization"] =
        water.EnthalpyOfVapourization(satTemp);

        // Calculate Source/Sink Term
        double sourceSink = norm * water.EnthalpyOfVapourization(satTemp) *
        water.Dw[i];
        boundaryRow["Source Sink Term"] = sourceSink;

        //Divide to Nodes
        if (M.CCode == 0)
        {
            lflx[M.Nd[i, 0]] = lflx[M.Nd[i, 0]] + (sourceSink / 2);
            lflx[M.Nd[i, 2]] = lflx[M.Nd[i, 2]] + (sourceSink / 2);
            boundaryRow["Node Value"] = sourceSink / 2;
        }

        else if (M.CCode == 1)
        {
            lflx[M.Nd[i, 0]] = lflx[M.Nd[i, 0]] + (sourceSink / 2) * 2 * Math.PI *
            M.nCd[M.Nd[i, 0], 0];
            lflx[M.Nd[i, 2]] = lflx[M.Nd[i, 2]] + (sourceSink / 2) * 2 * Math.PI *
            M.nCd[M.Nd[i, 2], 0];
            boundaryRow["Node Value"] = sourceSink / 2 * 2 * Math.PI * M.nCd[M.Nd[i,
            0], 0];
        }
    }
    #endregion

    #region // Line between element centroids neither horizontal nor vertical
    else
    {
        #region // Horizontal line between common nodes
        if (M.nCd[M.Nd[i, 0], 1] == M.nCd[M.Nd[i, 2], 1])
        {
            yz = M.nCd[M.Nd[i, 0], 1];
            m1 = (M.eCd[j, 1] - M.eCd[i, 1]) / (M.eCd[j, 0] - M.eCd[i, 0]);
            b1 = M.eCd[i, 1] - (m1 * M.eCd[i, 0]);
            x = (yz - b1) / m1;
            elmLength = Math.Abs(M.nCd[M.Nd[i, 0], 0] - M.nCd[M.Nd[i, 2], 0]);

            #region //Steam above liquid
            if (M.eCd[i, 1] > M.eCd[j, 1])
            {
                if (fluid.q[i, 1] < 0)
                {
```

```
                direction = 1;
                norm = Math.Abs(fluid.q[i, 1]) * direction * elmLength;
            }

            else if (fluid.q[i, 1] > 0)
            {
                direction = -1;
                norm = Math.Abs(fluid.q[i, 1]) * direction * elmLength;
            }

            else if (fluid.q[i, 1] == 0)
            {
                direction = 0;
                norm = Math.Abs(fluid.q[i, 1]) * direction * elmLength;
            }
        }
        #endregion

        #region //Liquid above steam
        else if (M.eCd[i, 1] < M.eCd[j, 1])
        {
            if (fluid.q[i, 1] < 0)
            {
                direction = -1;
                norm = Math.Abs(fluid.q[i, 1]) * direction * elmLength;
            }

            else if (fluid.q[i, 1] > 0)
            {
                direction = 1;
                norm = Math.Abs(fluid.q[i, 1]) * direction * elmLength;
            }

            else if (fluid.q[i, 1] == 0)
            {
                direction = 0;
                norm = Math.Abs(fluid.q[i, 1]) * direction * elmLength;
            }
        }
        #endregion

        boundaryRow["Code"] = "L";
    }
    #endregion

    #region // Vertical line between common nodes
    else if (M.nCd[M.Nd[i, 0], 0] == M.nCd[M.Nd[i, 2], 0])
    {
        x = M.nCd[M.Nd[i, 0], 0];
        m1 = (M.eCd[j, 1] - M.eCd[i, 1]) / (M.eCd[j, 0] - M.eCd[i, 0]);
        b1 = M.eCd[i, 1] - (m1 * M.eCd[i, 0]);
        yz = (m1 * x) + b1;
        elmLength = Math.Abs(M.nCd[M.Nd[i, 0], 1] - M.nCd[M.Nd[i, 2], 1]);

        #region //Steam right of liquid
        if (M.eCd[i, 0] > M.eCd[j, 0])
```

```
        {
            if (fluid.q[i, 0] < 0)
            {
                direction = 1;
                norm = Math.Abs(fluid.q[i, 0]) * direction * elmLength;
            }

            else if (fluid.q[i, 0] > 0)
            {
                direction = -1;
                norm = Math.Abs(fluid.q[i, 0]) * direction * elmLength;
            }

            else if (fluid.q[i, 0] == 0)
            {
                direction = 0;
                norm = Math.Abs(fluid.q[i, 0]) * direction * elmLength;
            }
        }
        #endregion

        #region //Liquid right of steam
        else if (M.eCd[i, 0] < M.eCd[j, 0])
        {
            if (fluid.q[i, 0] < 0)
            {
                direction = -1;
                norm = Math.Abs(fluid.q[i, 0]) * direction * elmLength;
            }

            else if (fluid.q[i, 0] > 0)
            {
                direction = 1;
                norm = Math.Abs(fluid.q[i, 0]) * direction * elmLength;
            }

            else if (fluid.q[i, 0] == 0)
            {
                direction = 0;
                norm = Math.Abs(fluid.q[i, 0]) * direction * elmLength;
            }
        }
        #endregion

        boundaryRow["Code"] = "M";
    }
    #endregion

    #region // Line between common nodes neither horizontal nor vertical
    else
    {
        m1 = (M.eCd[j, 1] - M.eCd[i, 1]) / (M.eCd[j, 0] - M.eCd[i, 0]);
        m2 = (M.nCd[M.Nd[i, 2], 1] - M.nCd[M.Nd[i, 0], 1]) / (M.nCd[M.Nd[i, 2], 0] -
        M.nCd[M.Nd[i, 0], 0]);
        b1 = M.eCd[i, 1] - (m1 * M.eCd[i, 0]);
        b2 = M.nCd[M.Nd[i, 0], 1] - (m2 * M.nCd[M.Nd[i, 0], 0]);
```

```
x = (b2 - b1) / (m1 - m2);
yz = (m1 * x) + b1;
elmLength = Math.Sqrt(((M.nCd[M.Nd[i, 0], 0] - M.nCd[M.Nd[i, 2], 0]) *
(M.nCd[M.Nd[i, 0], 0] - M.nCd[M.Nd[i, 2], 0])) + ((M.nCd[M.Nd[i, 0], 1] -
M.nCd[M.Nd[i, 2], 1]) * (M.nCd[M.Nd[i, 0], 1] - M.nCd[M.Nd[i, 2], 1])));
boundaryRow["Length"] = elmLength;

rotation = (Math.Atan((Math.Abs(M.nCd[M.Nd[i, 0], 0] - M.nCd[M.Nd[i, 2],
0])) / Math.Abs((M.nCd[M.Nd[i, 0], 1] - M.nCd[M.Nd[i, 2], 1]))));
boundaryRow["Theta"] = rotation;

#region //Negative slope for boundary
if (M.nCd[M.Nd[i, 0], 1] > M.nCd[M.Nd[i, 2], 1])
{
    #region //Steam above liquid
    if (M.eCd[i, 0] > M.eCd[j, 0])
    {
        direction = -1;
        norm = (fluid.q[i, 0] * Math.Cos(rotation) + fluid.q[i, 1] *
        Math.Sin(rotation)) * direction * elmLength;
        boundaryRow["Code"] = "N-S";
    }
    #endregion

    #region //Liquid above steam
    else if (M.eCd[i, 0] < M.eCd[j, 0])
    {
        direction = 1;
        norm = (fluid.q[i, 0] * Math.Cos(rotation) + fluid.q[i, 1] *
        Math.Sin(rotation)) * direction * elmLength;
        boundaryRow["Code"] = "N-L";
    }
    #endregion
}
#endregion

#region //Postive slope for boundary
else if (M.nCd[M.Nd[i, 0], 1] < M.nCd[M.Nd[i, 2], 1])
{
    #region //Steam above liquid
    if (M.eCd[i, 0] < M.eCd[j, 0])
    {
        direction = -1;
        norm = (fluid.q[i, 0] * Math.Cos(Math.PI - rotation) + fluid.q[i, 1] *
        Math.Sin(Math.PI - rotation)) * direction * elmLength;
        boundaryRow["Code"] = "N+S";
    }
    #endregion

    #region //Liquid above steam
    else if (M.eCd[i, 0] > M.eCd[j, 0])
    {
        direction = 1;
        norm = (fluid.q[i, 0] * Math.Cos(Math.PI - rotation) + fluid.q[i, 1] *
        Math.Sin(Math.PI - rotation)) * direction * elmLength;
        boundaryRow["Code"] = "N+L";
```

```
                }
                #endregion
            }
            #endregion
        }
        #endregion

        //Flux values
        boundaryRow["Fluid Flux"] = Math.Sqrt((fluid.q[i, 0] * fluid.q[i, 0]) + (fluid.q[i, 1]
        * fluid.q[i, 1]));
        boundaryRow["Normal Flux"] = norm;

        // Interpolate for saturation temperature
        disti = Math.Sqrt(((yz - M.eCd[i, 1]) * (yz - M.eCd[i, 1])) + ((x - M.eCd[i, 0]) * (x -
        M.eCd[i, 0])));
        distj = Math.Sqrt(((yz - M.eCd[j, 1]) * (yz - M.eCd[j, 1])) + ((x - M.eCd[j, 0]) * (x -
        M.eCd[j, 0])));

        double satTemp = water.Tf[i] + ((0 - (-disti)) / (distj - (-disti))) * (water.Tf[j] -
        water.Tf[i]);

        boundaryRow["Transition Temperature"] = satTemp;
        boundaryRow["Enthalpy Vaporization"] =
        water.EnthalpyOfVapourization(satTemp);

        //Calculate Source/Sink Term
        double sourceSink = norm * water.EnthalpyOfVapourization(satTemp) *
        water.Dw[i];

        boundaryRow["Source Sink Term"] = sourceSink;

        //Divide to Nodes
        if (M.CCode == 0)
        {
            lflx[M.Nd[i, 0]] = lflx[M.Nd[i, 0]] + (sourceSink / 2);
            lflx[M.Nd[i, 2]] = lflx[M.Nd[i, 2]] + (sourceSink / 2);
            boundaryRow["Node Value"] = sourceSink / 2;
        }

        else if (M.CCode == 1)
        {
            lflx[M.Nd[i, 0]] = lflx[M.Nd[i, 0]] + (sourceSink / 2) * 2 * Math.PI *
            M.nCd[M.Nd[i, 0], 0];
            lflx[M.Nd[i, 2]] = lflx[M.Nd[i, 2]] + (sourceSink / 2) * 2 * Math.PI *
            M.nCd[M.Nd[i, 2], 0];
            boundaryRow["Node Value"] = sourceSink / 2 * 2 * Math.PI * M.nCd[M.Nd[i,
            0], 0];
        }
    }
    #endregion

    steamBoundary.Rows.Add(boundaryRow);
}

// The second (1) and third (2) nodes from the steam element are shared with two of the nodes
from the liquid element
```

```
else if (((M.Nd[i, 1] == M.Nd[j, 0]) || (M.Nd[i, 1] == M.Nd[j, 1]) || (M.Nd[i, 1] == M.Nd[j,
2])) && ((M.Nd[i, 2] == M.Nd[j, 0]) || (M.Nd[i, 2] == M.Nd[j, 1]) || (M.Nd[i, 2] == M.Nd[j,
2])))
{
    DataRow boundaryRow = steamBoundary.NewRow();
    boundaryRow["Node 1"] = M.Nd[i, 1];
    boundaryRow["Node 2"] = M.Nd[i, 2];
    boundaryRow["Steam Element"] = i;
    boundaryRow["Liquid Element"] = j;
    boundaryRow["Steam Temperature"] = water.Tf[i];
    boundaryRow["Liquid Temperature"] = water.Tf[j];
    boundaryRow["Flux X"] = fluid.q[i, 0];
    boundaryRow["Flux Y"] = fluid.q[i, 1];
    boundaryRow["Density"] = water.Dw[i];

    double x, yz, m1, m2, b1, b2, disti, distj, rotation, elmLength;
    double norm = 0;
    int direction;

    #region // Vertical line between element centroids
    if (M.eCd[i, 0] == M.eCd[j, 0])
    {
        #region // Horizontal line between common nodes
        if (M.nCd[M.Nd[i, 1], 1] == M.nCd[M.Nd[i, 2], 1])
        {
            x = M.eCd[i, 0];
            yz = M.nCd[M.Nd[i, 1], 1];
            elmLength = Math.Abs(M.nCd[M.Nd[i, 1], 0] - M.nCd[M.Nd[i, 2], 0]);

            #region //Steam above liquid
            if (M.eCd[i, 1] > M.eCd[j, 1])
            {
                if (fluid.q[i, 1] < 0)
                {
                    direction = 1;
                    norm = Math.Abs(fluid.q[i, 1]) * direction * elmLength;
                }

                else if (fluid.q[i, 1] > 0)
                {
                    direction = -1;
                    norm = Math.Abs(fluid.q[i, 1]) * direction * elmLength;
                }

                else if (fluid.q[i, 1] == 0)
                {
                    direction = 0;
                    norm = Math.Abs(fluid.q[i, 1]) * direction * elmLength;
                }
            }
            #endregion

            #region //Liquid above steam
            else if (M.eCd[i, 1] < M.eCd[j, 1])
            {
                if (fluid.q[i, 1] < 0)
```

```
            {
                direction = -1;
                norm = Math.Abs(fluid.q[i, 1]) * direction * elmLength;
            }

            else if (fluid.q[i, 1] > 0)
            {
                direction = 1;
                norm = Math.Abs(fluid.q[i, 1]) * direction * elmLength;
            }

            else if (fluid.q[i, 1] == 0)
            {
                direction = 0;
                norm = Math.Abs(fluid.q[i, 1]) * direction * elmLength;
            }
        }
        #endregion

        boundaryRow["Code"] = "O";
    }
    #endregion

    #region // Line between common nodes neither horizontal nor vertical
    else
    {
        x = M.eCd[i, 0];
        m2 = (M.nCd[M.Nd[i, 2], 1] - M.nCd[M.Nd[i, 1], 1]) / (M.nCd[M.Nd[i, 2], 0] -
        M.nCd[M.Nd[i, 1], 0]);
        b2 = M.nCd[M.Nd[i, 1], 1] - (m2 * M.nCd[M.Nd[i, 1], 0]);
        yz = (m2 * x) + b2;
        elmLength = Math.Sqrt(((M.nCd[M.Nd[i, 1], 0] - M.nCd[M.Nd[i, 2], 0]) *
        (M.nCd[M.Nd[i, 1], 0] - M.nCd[M.Nd[i, 2], 0])) + ((M.nCd[M.Nd[i, 1], 1] -
        M.nCd[M.Nd[i, 2], 1]) * (M.nCd[M.Nd[i, 1], 1] - M.nCd[M.Nd[i, 2], 1])));
        boundaryRow["Length"] = elmLength;

        rotation = (Math.Atan((Math.Abs(M.nCd[M.Nd[i, 1], 0] - M.nCd[M.Nd[i, 2],
        0])) / Math.Abs((M.nCd[M.Nd[i, 1], 1] - M.nCd[M.Nd[i, 2], 1]))));
        boundaryRow["Theta"] = rotation;

        #region //Negative slope for boundary
        if (M.nCd[M.Nd[i, 1], 1] > M.nCd[M.Nd[i, 2], 1])
        {
            #region //Steam above liquid
            if (M.eCd[i, 1] > M.eCd[j, 1])
            {
                direction = -1;
                norm = (fluid.q[i, 0] * Math.Cos(rotation) + fluid.q[i, 1] *
                Math.Sin(rotation)) * direction * elmLength;
                boundaryRow["Code"] = "P-S";
            }
            #endregion

            #region //Liquid above steam
            else if (M.eCd[i, 1] < M.eCd[j, 1])
            {
```

```
                direction = 1;
                norm = (fluid.q[i, 0] * Math.Cos(rotation) + fluid.q[i, 1] *
                Math.Sin(rotation)) * direction * elmLength;
                boundaryRow["Code"] = "P-L";
            }
            #endregion
        }
        #endregion

        #region //Postive slope for boundary
        else if (M.nCd[M.Nd[i, 1], 1] < M.nCd[M.Nd[i, 2], 1])
        {
            #region //Steam above liquid
            if (M.eCd[i, 1] > M.eCd[j, 1])
            {
                direction = -1;
                norm = (fluid.q[i, 0] * Math.Cos(Math.PI - rotation) + fluid.q[i, 1] *
                Math.Sin(Math.PI - rotation)) * direction * elmLength;
                boundaryRow["Code"] = "P+S";
            }
            #endregion

            #region //Liquid above steam
            else if (M.eCd[i, 1] < M.eCd[j, 1])
            {
                direction = 1;
                norm = (fluid.q[i, 0] * Math.Cos(Math.PI - rotation) + fluid.q[i, 1] *
                Math.Sin(Math.PI - rotation)) * direction * elmLength;
                boundaryRow["Code"] = "P+L";
            }
            #endregion
        }
        #endregion
    }
    #endregion

    //Flux values
    boundaryRow["Fluid Flux"] = Math.Sqrt((fluid.q[i, 0] * fluid.q[i, 0]) + (fluid.q[i, 1]
    * fluid.q[i, 1]));
    boundaryRow["Normal Flux"] = norm;

    // Interpolate for saturation temperature
    disti = Math.Sqrt(((yz - M.eCd[i, 1]) * (yz - M.eCd[i, 1])) + ((x - M.eCd[i, 0]) * (x -
    M.eCd[i, 0])));
    distj = Math.Sqrt(((yz - M.eCd[j, 1]) * (yz - M.eCd[j, 1])) + ((x - M.eCd[j, 0]) * (x -
    M.eCd[j, 0])));

    double satTemp = water.Tf[i] + ((0 - (-disti)) / (distj - (-disti))) * (water.Tf[j] -
    water.Tf[i]);

    boundaryRow["Transition Temperature"] = satTemp;
    boundaryRow["Enthalpy Vaporization"] =
    water.EnthalpyOfVapourization(satTemp);

    //Calculate Source/Sink Term
```

```
        double sourceSink = norm * water.EnthalpyOfVapourization(satTemp) *
        water.Dw[i];

        boundaryRow["Source Sink Term"] = sourceSink;

        //Divide to Nodes
        if (M.CCode == 0)
        {
            lflx[M.Nd[i, 1]] = lflx[M.Nd[i, 1]] + (sourceSink / 2);
            lflx[M.Nd[i, 2]] = lflx[M.Nd[i, 2]] + (sourceSink / 2);
            boundaryRow["Node Value"] = sourceSink / 2;
        }

        else if (M.CCode == 1)
        {
            lflx[M.Nd[i, 1]] = lflx[M.Nd[i, 1]] + (sourceSink / 2) * 2 * Math.PI *
            M.nCd[M.Nd[i, 1], 0];
            lflx[M.Nd[i, 2]] = lflx[M.Nd[i, 2]] + (sourceSink / 2) * 2 * Math.PI *
            M.nCd[M.Nd[i, 2], 0];
            boundaryRow["Node Value"] = sourceSink / 2 * 2 * Math.PI * M.nCd[M.Nd[i,
            1], 0];
        }
    }
    #endregion

    #region // Horizontal line between element centroids

    else if (M.eCd[i, 1] == M.eCd[j, 1])
     {
        #region // Vertical line between common nodes
        if (M.nCd[M.Nd[i, 1], 0] == M.nCd[M.Nd[i, 2], 0])
        {
            x = M.nCd[M.Nd[i, 1], 0];
            yz = M.eCd[i, 1];
            elmLength = Math.Abs(M.nCd[M.Nd[i, 1], 1] - M.nCd[M.Nd[i, 2], 1]);

            #region //Steam right of liquid
            if (M.eCd[i, 0] > M.eCd[j, 0])
            {
                if (fluid.q[i, 0] < 0)
                {
                    direction = 1;
                    norm = Math.Abs(fluid.q[i, 0]) * direction * elmLength;
                }

                else if (fluid.q[i, 0] > 0)
                {
                    direction = -1;
                    norm = Math.Abs(fluid.q[i, 0]) * direction * elmLength;
                }

                else if (fluid.q[i, 0] == 0)
                {
                    direction = 0;
                    norm = Math.Abs(fluid.q[i, 0]) * direction * elmLength;
                }
```

```
        }
        #endregion

        #region //Liquid right of steam
        else if (M.eCd[i, 0] < M.eCd[j, 0])
        {
            if (fluid.q[i, 0] < 0)
            {
                direction = -1;
                norm = Math.Abs(fluid.q[i, 0]) * direction * elmLength;
            }

            else if (fluid.q[i, 0] > 0)
            {
                direction = 1;
                norm = Math.Abs(fluid.q[i, 0]) * direction * elmLength;
            }

            else if (fluid.q[i, 0] == 0)
            {
                direction = 0;
                norm = Math.Abs(fluid.q[i, 0]) * direction * elmLength;
            }
        }
        #endregion

        boundaryRow["Code"] = "Q";
    }
    #endregion

    #region // Line between common nodes neither horizontal nor vertical
    else
    {
        yz = M.eCd[i, 1];
        m2 = (M.nCd[M.Nd[i, 2], 1] - M.nCd[M.Nd[i, 1], 1]) / (M.nCd[M.Nd[i, 2], 0] -
        M.nCd[M.Nd[i, 1], 0]);
        b2 = M.nCd[M.Nd[i, 1], 1] - (m2 * M.nCd[M.Nd[i, 1], 0]);
        x = (yz - b2) / m2;
        elmLength = Math.Sqrt(((M.nCd[M.Nd[i, 1], 0] - M.nCd[M.Nd[i, 2], 0]) *
        M.nCd[M.Nd[i, 1], 0] - M.nCd[M.Nd[i, 2], 0])) + ((M.nCd[M.Nd[i, 1], 1] -
        M.nCd[M.Nd[i, 2], 1]) * (M.nCd[M.Nd[i, 1], 1] - M.nCd[M.Nd[i, 2], 1])));
        boundaryRow["Length"] = elmLength;

        rotation = (Math.Atan((Math.Abs(M.nCd[M.Nd[i, 1], 0] - M.nCd[M.Nd[i, 2],
        0])) / Math.Abs((M.nCd[M.Nd[i, 1], 1] - M.nCd[M.Nd[i, 2], 1]))));
        boundaryRow["Theta"] = rotation;

        #region //Negative slope for boundary
        if (M.nCd[M.Nd[i, 1], 1] > M.nCd[M.Nd[i, 2], 1])
        {
            #region //Steam above liquid
            if (M.eCd[i, 0] > M.eCd[j, 0])
            {
                direction = -1;
                norm = (fluid.q[i, 0] * Math.Cos(rotation) + fluid.q[i, 1] *
                Math.Sin(rotation)) * direction * elmLength;
```

```
                boundaryRow["Code"] = "R-S";
            }
            #endregion

            #region //Liquid above steam
            else if (M.eCd[i, 0] < M.eCd[j, 0])
            {
                direction = 1;
                norm = (fluid.q[i, 0] * Math.Cos(rotation) + fluid.q[i, 1] *
                Math.Sin(rotation)) * direction * elmLength;
                boundaryRow["Code"] = "R-L";
            }
            #endregion
        }
        #endregion

        #region //Postive slope for boundary
        else if (M.nCd[M.Nd[i, 1], 1] < M.nCd[M.Nd[i, 2], 1])
        {
            #region //Steam above liquid
            if (M.eCd[i, 0] < M.eCd[j, 0])
            {
                direction = -1;
                norm = (fluid.q[i, 0] * Math.Cos(Math.PI - rotation) + fluid.q[i, 1] *
                Math.Sin(Math.PI - rotation)) * direction * elmLength;
                boundaryRow["Code"] = "R+S";
            }
            #endregion

            #region //Liquid above steam
            else if (M.eCd[i, 0] > M.eCd[j, 0])
            {
                direction = 1;
                norm = (fluid.q[i, 0] * Math.Cos(Math.PI - rotation) + fluid.q[i, 1] *
                Math.Sin(Math.PI - rotation)) * direction * elmLength;
                boundaryRow["Code"] = "R+L";
            }
            #endregion
        }
        #endregion
    }
    #endregion

    //Flux values
    boundaryRow["Fluid Flux"] = Math.Sqrt((fluid.q[i, 0] * fluid.q[i, 0]) + (fluid.q[i, 1]
    * fluid.q[i, 1]));
    boundaryRow["Normal Flux"] = norm;

    // Interpolate for saturation pressure
    disti = Math.Sqrt(((yz - M.eCd[i, 1]) * (yz - M.eCd[i, 1])) + ((x - M.eCd[i, 0]) * (x -
    M.eCd[i, 0])));
    distj = Math.Sqrt(((yz - M.eCd[j, 1]) * (yz - M.eCd[j, 1])) + ((x - M.eCd[j, 0]) * (x -
    M.eCd[j, 0])));

    double satTemp = water.Tf[i] + ((0 - (-disti)) / (distj - (-disti))) * (water.Tf[j] -
    water.Tf[i]);
```

```
        boundaryRow["Transition Temperature"] = satTemp;
        boundaryRow["Enthalpy Vaporization"] =
        water.EnthalpyOfVapourization(satTemp);

        //Calculate Source/Sink Term
        double sourceSink = norm * water.EnthalpyOfVapourization(satTemp) *
        water.Dw[i];

        boundaryRow["Source Sink Term"] = sourceSink;

        //Divide to Nodes
        if (M.CCode == 0)
        {
            lflx[M.Nd[i, 1]] = lflx[M.Nd[i, 1]] + (sourceSink / 2);
            lflx[M.Nd[i, 2]] = lflx[M.Nd[i, 2]] + (sourceSink / 2);
            boundaryRow["Node Value"] = sourceSink / 2;
        }

        else if (M.CCode == 1)
        {
            lflx[M.Nd[i, 1]] = lflx[M.Nd[i, 1]] + (sourceSink / 2) * 2 * Math.PI *
            M.nCd[M.Nd[i, 1], 0];
            lflx[M.Nd[i, 2]] = lflx[M.Nd[i, 2]] + (sourceSink / 2) * 2 * Math.PI *
            M.nCd[M.Nd[i, 2], 0];
            boundaryRow["Node Value"] = sourceSink / 2 * 2 * Math.PI * M.nCd[M.Nd[i,
            1], 0];
        }
    }
    #endregion

    #region // Line between element centroids neither horizontal nor vertical
    else
    {
        #region // Horizontal line between common nodes
        if (M.nCd[M.Nd[i, 1], 1] == M.nCd[M.Nd[i, 2], 1])
        {
            yz = M.nCd[M.Nd[i, 1], 1];
            m1 = (M.eCd[j, 1] - M.eCd[i, 1]) / (M.eCd[j, 0] - M.eCd[i, 0]);
            b1 = M.eCd[i, 1] - (m1 * M.eCd[i, 0]);
            x = (yz - b1) / m1;
            elmLength = Math.Abs(M.nCd[M.Nd[i, 1], 0] - M.nCd[M.Nd[i, 2], 0]);

            #region //Steam above liquid
            if (M.eCd[i, 1] > M.eCd[j, 1])
            {
                if (fluid.q[i, 1] < 0)
                {
                    direction = 1;
                    norm = Math.Abs(fluid.q[i, 1]) * direction * elmLength;
                }

                else if (fluid.q[i, 1] > 0)
                {
                    direction = -1;
                    norm = Math.Abs(fluid.q[i, 1]) * direction * elmLength;
```

```
            }

            else if (fluid.q[i, 1] == 0)
            {
                direction = 0;
                norm = Math.Abs(fluid.q[i, 1]) * direction * elmLength;
            }
        }
        #endregion

        #region //Liquid above steam
        else if (M.eCd[i, 1] < M.eCd[j, 1])
        {
            if (fluid.q[i, 1] < 0)
            {
                direction = -1;
                norm = Math.Abs(fluid.q[i, 1]) * direction * elmLength;
            }

            else if (fluid.q[i, 1] > 0)
            {
                direction = 1;
                norm = Math.Abs(fluid.q[i, 1]) * direction * elmLength;
            }

            else if (fluid.q[i, 1] == 0)
            {
                direction = 0;
                norm = Math.Abs(fluid.q[i, 1]) * direction * elmLength;
            }
        }
        #endregion

        boundaryRow["Code"] = "S";
    }
    #endregion

    #region // Vertical line between common nodes
    else if (M.nCd[M.Nd[i, 1], 0] == M.nCd[M.Nd[i, 2], 0])
    {
        x = M.nCd[M.Nd[i, 1], 0];
        m1 = (M.eCd[j, 1] - M.eCd[i, 1]) / (M.eCd[j, 0] - M.eCd[i, 0]);
        b1 = M.eCd[i, 1] - (m1 * M.eCd[i, 0]);
        yz = (m1 * x) + b1;
        elmLength = Math.Abs(M.nCd[M.Nd[i, 1], 1] - M.nCd[M.Nd[i, 2], 1]);

        #region //Steam right of liquid
        if (M.eCd[i, 0] > M.eCd[j, 0])
        {
            if (fluid.q[i, 0] < 0)
            {
                direction = 1;
                norm = Math.Abs(fluid.q[i, 0]) * direction * elmLength;
            }

            else if (fluid.q[i, 0] > 0)
```

```
            {
                direction = -1;
                norm = Math.Abs(fluid.q[i, 0]) * direction * elmLength;
            }

            else if (fluid.q[i, 0] == 0)
            {
                direction = 0;
                norm = Math.Abs(fluid.q[i, 0]) * direction * elmLength;
            }
        }
        #endregion

        #region //Liquid right of steam
        else if (M.eCd[i, 0] < M.eCd[j, 0])
        {
            if (fluid.q[i, 0] < 0)
            {
                direction = -1;
                norm = Math.Abs(fluid.q[i, 0]) * direction * elmLength;
            }

            else if (fluid.q[i, 0] > 0)
            {
                direction = 1;
                norm = Math.Abs(fluid.q[i, 0]) * direction * elmLength;
            }

            else if (fluid.q[i, 0] == 0)
            {
                direction = 0;
                norm = Math.Abs(fluid.q[i, 0]) * direction * elmLength;
            }
        }
        #endregion

        boundaryRow["Code"] = "T";
    }
    #endregion

    #region // Line between common nodes neither horizontal nor vertical
    else
    {
        m1 = (M.eCd[j, 1] - M.eCd[i, 1]) / (M.eCd[j, 0] - M.eCd[i, 0]);
        m2 = (M.nCd[M.Nd[i, 2], 1] - M.nCd[M.Nd[i, 1], 1]) / (M.nCd[M.Nd[i, 2], 0] -
        M.nCd[M.Nd[i, 1], 0]);
        b1 = M.eCd[i, 1] - (m1 * M.eCd[i, 0]);
        b2 = M.nCd[M.Nd[i, 1], 1] - (m2 * M.nCd[M.Nd[i, 1], 0]);
        x = (b2 - b1) / (m1 - m2);
        yz = (m1 * x) + b1;
        elmLength = Math.Sqrt(((M.nCd[M.Nd[i, 1], 0] - M.nCd[M.Nd[i, 2], 0]) *
        (M.nCd[M.Nd[i, 1], 0] - M.nCd[M.Nd[i, 2], 0])) + ((M.nCd[M.Nd[i, 1], 1] -
        M.nCd[M.Nd[i, 2], 1]) * (M.nCd[M.Nd[i, 1], 1] - M.nCd[M.Nd[i, 2], 1])));
        boundaryRow["Length"] = elmLength;
```

```
            rotation = (Math.Atan((Math.Abs(M.nCd[M.Nd[i, 1], 0] - M.nCd[M.Nd[i, 2],
            0])) / Math.Abs((M.nCd[M.Nd[i, 1], 1] - M.nCd[M.Nd[i, 2], 1]))));
            boundaryRow["Theta"] = rotation;

            #region //Negative slope for boundary
            if (M.nCd[M.Nd[i, 1], 1] > M.nCd[M.Nd[i, 2], 1])
            {
                #region //Steam above liquid
                if (M.eCd[i, 0] > M.eCd[j, 0])
                {
                    direction = -1;
                    norm = (fluid.q[i, 0] * Math.Cos(rotation) + fluid.q[i, 1] *
                    Math.Sin(rotation)) * direction * elmLength;
                    boundaryRow["Code"] = "U-S";
                }
                #endregion

                #region //Liquid above steam
                else if (M.eCd[i, 0] < M.eCd[j, 0])
                {
                    direction = 1;
                    norm = (fluid.q[i, 0] * Math.Cos(rotation) + fluid.q[i, 1] *
                    Math.Sin(rotation)) * direction * elmLength;
                    boundaryRow["Code"] = "U-L";
                }
                #endregion
            }
            #endregion

            #region //Postive slope for boundary
            else if (M.nCd[M.Nd[i, 1], 1] < M.nCd[M.Nd[i, 2], 1])
            {
                #region //Steam above liquid
                if (M.eCd[i, 0] < M.eCd[j, 0])
                {
                    direction = -1;
                    norm = (fluid.q[i, 0] * Math.Cos(Math.PI - rotation) + fluid.q[i, 1] *
                    Math.Sin(Math.PI - rotation)) * direction * elmLength;
                    boundaryRow["Code"] = "U+S";
                }
                #endregion

                #region //Liquid above steam
                else if (M.eCd[i, 0] > M.eCd[j, 0])
                {
                    direction = 1;
                    norm = (fluid.q[i, 0] * Math.Cos(Math.PI - rotation) + fluid.q[i, 1] *
                    Math.Sin(Math.PI - rotation)) * direction * elmLength;
                    boundaryRow["Code"] = "U+L";
                }
                #endregion
            }
            #endregion
        }
        #endregion
```

```
                    //Flux values
                    boundaryRow["Fluid Flux"] = Math.Sqrt((fluid.q[i, 0] * fluid.q[i, 0]) + (fluid.q[i, 1]
                    * fluid.q[i, 1]));
                    boundaryRow["Normal Flux"] = norm;

                    // Interpolate for saturation pressure
                    disti = Math.Sqrt(((yz - M.eCd[i, 1]) * (yz - M.eCd[i, 1])) + ((x - M.eCd[i, 0]) * (x -
                    M.eCd[i, 0])));
                    distj = Math.Sqrt(((yz - M.eCd[j, 1]) * (yz - M.eCd[j, 1])) + ((x - M.eCd[j, 0]) * (x -
                    M.eCd[j, 0])));

                    double satTemp = water.Tf[i] + ((0 - (-disti)) / (distj - (-disti))) * (water.Tf[j] -
                    water.Tf[i]);

                    boundaryRow["Transition Temperature"] = satTemp;
                    boundaryRow["Enthalpy Vaporization"] =
                    water.EnthalpyOfVapourization(satTemp);

                    //Calculate Source/Sink Term
                    double sourceSink = norm * water.EnthalpyOfVapourization(satTemp) *
                    water.Dw[i];

                    boundaryRow["Source Sink Term"] = sourceSink;

                    //Divide to Nodes
                    if (M.CCode == 0)
                    {
                        lflx[M.Nd[i, 1]] = lflx[M.Nd[i, 1]] + (sourceSink / 2);
                        lflx[M.Nd[i, 2]] = lflx[M.Nd[i, 2]] + (sourceSink / 2);
                        boundaryRow["Node Value"] = sourceSink / 2;
                    }

                    else if (M.CCode == 1)
                    {
                        lflx[M.Nd[i, 1]] = lflx[M.Nd[i, 1]] + (sourceSink / 2) * 2 * Math.PI *
                        M.nCd[M.Nd[i, 1], 0];
                        lflx[M.Nd[i, 2]] = lflx[M.Nd[i, 2]] + (sourceSink / 2) * 2 * Math.PI *
                        M.nCd[M.Nd[i, 2], 0];
                        boundaryRow["Node Value"] = sourceSink / 2 * 2 * Math.PI * M.nCd[M.Nd[i,
                        1], 0];
                    }
                }
                #endregion

                steamBoundary.Rows.Add(boundaryRow);
            }
        }
    }
}

#region // When steam boundary is system boundary - left side
for (int i = 0; i < M.nElm; i++)
{
    if (water.pReg[i] == 3)
    {
        for (int j = M.bNd[0, 0]; j < M.bNd[0, 1]; j++)
```

```
            {
                if ((M.Nd[i, 0] == j) && (M.Nd[i, 1] == j + 1))
                {
                    DataRow boundaryRow = steamBoundary.NewRow();
                    boundaryRow["Node 1"] = M.Nd[i, 0];
                    boundaryRow["Node 2"] = M.Nd[i, 1];
                    boundaryRow["Steam Element"] = i;
                    steamBoundary.Rows.Add(boundaryRow);
                }
            }
        }
    }
#endregion

#region // When steam boundary is system boundary - right side
for (int i = 0; i < M.nElm; i++)
{
    if (water.pReg[i] == 3)
    {
        for (int j = M.bNd[M.cMax, 0]; j < M.bNd[M.cMax, 1]; j++)
        {
            if ((M.Nd[i, 2] == j) && (M.Nd[i, 1] == j + 1))
            {
                DataRow boundaryRow = steamBoundary.NewRow();
                boundaryRow["Node 1"] = M.Nd[i, 2];
                boundaryRow["Node 2"] = M.Nd[i, 1];
                boundaryRow["Steam Element"] = i;
                steamBoundary.Rows.Add(boundaryRow);
            }
        }
    }
}
#endregion

#region // When steam boundary is system boundary - upper boundary
for (int i = 0; i < M.nElm; i++)
{
    if (water.pReg[i] == 3)
    {
        for (int j = 0; j < M.cMax; j++)
        {
            if ((M.Nd[i, 0] == M.bNd[j, 0]) && (M.Nd[i, 2]) == M.bNd[j + 1, 0])
            {
                DataRow boundaryRow = steamBoundary.NewRow();
                boundaryRow["Node 1"] = M.Nd[i, 0];
                boundaryRow["Node 2"] = M.Nd[i, 2];
                boundaryRow["Steam Element"] = i;
                steamBoundary.Rows.Add(boundaryRow);
            }
        }
    }
}
#endregion

#region // When steam boundary is system boundary - lower boundary
```

```
for (int i = 0; i < M.nElm; i++)
{
    if (water.pReg[i] == 3)
    {
        for (int j = 0; j < M.cMax; j++)
        {
            if ((M.Nd[i, 0] == M.bNd[j, 1]) && (M.Nd[i, 1]) == M.bNd[j + 1, 1])
            {
                DataRow boundaryRow = steamBoundary.NewRow();
                boundaryRow["Node 1"] = M.Nd[i, 0];
                boundaryRow["Node 2"] = M.Nd[i, 1];
                boundaryRow["Steam Element"] = i;
                steamBoundary.Rows.Add(boundaryRow);
            }

            else if ((M.Nd[i, 1] == M.bNd[j, 1]) && (M.Nd[i, 2]) == M.bNd[j + 1, 1])
            {
                DataRow boundaryRow = steamBoundary.NewRow();
                boundaryRow["Node 1"] = M.Nd[i, 1];
                boundaryRow["Node 2"] = M.Nd[i, 2];
                boundaryRow["Steam Element"] = i;
                steamBoundary.Rows.Add(boundaryRow);
            }
        }
    }
}
#endregion

string outFile = "SteamData.adf";

FileStream fs = new FileStream(outFile, FileMode.Create, FileAccess.Write);
GZipStream cs = new GZipStream(fs, CompressionMode.Compress);
StreamWriter sw = new StreamWriter(cs);

sw.WriteLine("{0}, {1}, {2}, {3}, {4}, {5}, {6}, {7}, {8}, {9}, {10}, {11}, {12}, {13}", "Code, ", "Node 1, ", "Node 2, ", "Steam Element, ", "Liquid Element, ", "Fluid Flux, ", "Flux X, ", "Flux Y, ", "Normal Flux, ", "Density, ", "Enthalpy Vaporization, ", "Source Sink Term, ", "Length, ", "Theta, ");

foreach (DataRow r in steamBoundary.Rows)
{
    sw.WriteLine("{0}, {1}, {2}, {3}, {4}, {5}, {6}, {7}, {8}, {9}, {10}, {11}, {12}, {13}", r["Code"],
    r["Node 1"], r["Node 2"], r["Steam Element"], r["Liquid Element"], r["Fluid Flux"], r["Flux X"],
    r["Flux Y"], r["Normal Flux"], r["Density"], r["Enthalpy Vaporization"], r["Source Sink Term"],
    r["Length"], r["Theta"]);
}

sw.Close();
cs.Close();
fs.Close();
}
```

APPENDIX 5: EFFECTS OF LATENT HEAT ON STEAM BOUNDARY LOCATION

Each combination of pressures, permeabilities, and thermal gradients was run twice: once without the latent heat algorithm and once with the latent heat algorithm. The location of the liquid-steam boundary in the column was noted and compared for each pair of models, yielding a displacement value. This value was used to quantify the effects of the latent heat or the magnitude of the "shift" of the boundary. Displacement values are listed in Tables A5.1 (a – c) and A5.2 (a – c).

Table A5.1: Displacement values for pairs of models indicating the magnitude of the effect of latent heat on the liquid-steam boundary for down temperature flow. a) 6 MPa b) 12 MPa c) 18 MPa

Table A5.1 a) Down Temperature Flow			
Pressure (MPa)	**Thermal Gradient (°C/m)**	**Permeability (m^2)**	**Displacement (m)**
6	0.05	10^{-18}	363.35
		10^{-17}	683.22
		10^{-16}	189.35
		10^{-15}	7.68
		10^{-14}	0
	0.10	10^{-18}	509.22
		10^{-17}	614.13
		10^{-16}	79.33
		10^{-15}	7.68
		10^{-14}	0
	0.15	10^{-18}	475.95
		10^{-17}	481.06
		10^{-16}	37.71
		10^{-15}	5.12
		10^{-14}	0

Table A5.1 b) Down Temperature Flow			
Pressure (MPa)	**Thermal Gradient (°C/m)**	**Permeability (m^2)**	**Displacement (m)**
12	0.05	10^{-18}	307.16
		10^{-17}	639.71
		10^{-16}	209.83
		10^{-15}	23.03
		10^{-14}	0
	0.10	10^{-18}	404.3
		10^{-17}	624.36
		10^{-16}	97.24
		10^{-15}	5.12
		10^{-14}	0
	0.15	10^{-18}	258.44
		10^{-17}	588.53
		10^{-16}	48.62
		10^{-15}	7.68
		10^{-14}	0

Table A5.1 c) Down Temperature Flow			
Pressure (MPa)	**Thermal Gradient (°C/m)**	**Permeability (m^2)**	**Displacement (m)**
18	0.05	10^{-18}	115.15
		10^{-17}	240.53
		10^{-16}	148.42
		10^{-15}	14.47
		10^{-14}	0
	0.10	10^{-18}	199.59
		10^{-17}	299.39
		10^{-16}	92.12
		10^{-15}	10.24
		10^{-14}	0
	0.15	10^{-18}	99.8
		10^{-17}	337.77
		10^{-16}	48.62
		10^{-15}	5.11
		10^{-14}	0

Table A5.2: Displacement values for pairs of models indicating the magnitude of the effect of latent heat on the liquid-steam boundary for up temperature flow. a) 6 MPa b) 12 MPa c) 18 MPa

Table A6.5 a) Up Temperature Flow			
Pressure (MPa)	**Thermal Gradient (°C/m)**	**Permeability (m^2)**	**Displacement (m)**
12	0.05	10^{-18}	629.48
		10^{-17}	1322.92
		10^{-16}	23.03
		10^{-15}	5.11
		10^{-14}	0
	0.10	10^{-18}	122.82
		10^{-17}	284.04
		10^{-16}	51.17
		10^{-15}	5.12
		10^{-14}	0
	0.15	10^{-18}	81.88
		10^{-17}	46.05
		10^{-16}	23.03
		10^{-15}	2.56
		10^{-14}	0

Table A6.5 b) Up Temperature Flow			
Pressure (MPa)	**Thermal Gradient (°C/m)**	**Permeability (m^2)**	**Displacement (m)**
12	0.05	10^{-18}	140.73
		10^{-17}	452.46
		10^{-16}	156.09
		10^{-15}	7.68
		10^{-14}	0
	0.10	10^{-18}	51.18
		10^{-17}	92.12
		10^{-16}	43.5
		10^{-15}	7.68
		10^{-14}	0
	0.15	10^{-18}	40.94
		10^{-17}	17.91
		10^{-16}	15.35
		10^{-15}	2.56
		10^{-14}	0

Table A6.5 c) Up Temperature Flow			
Pressure (MPa)	**Thermal Gradient (°C/m)**	**Permeability (m^2)**	**Displacement (m)**
18	0.05	10^{-18}	330.1
		10^{-17}	972.36
		10^{-16}	61.41
		10^{-15}	10.24
		10^{-14}	0
	0.10	10^{-18}	53.73
		10^{-17}	117.7
		10^{-16}	69.08
		10^{-15}	7.68
		10^{-14}	0
	0.15	10^{-18}	33.27
		10^{-17}	51.18
		10^{-16}	28.15
		10^{-15}	10.23
		10^{-14}	0